Septel™ Product Range

SS7 Programmer's Manual for Septel ISA (PCCS6)

DataKinetics Ltd Fordingbridge Hants England Tel: +44 1425 651300 Fax: +44 1425 655075

IMPORTANT INFORMATION

The information in this manual is supplied without warranty as to its accuracy. DataKinetics is not responsible or liable for any loss or damage of whatever kind arising from the use of the Septel ISA (PCCS6), PCCS3 or the supporting software and documentation.

©**1993-2001 DataKinetics Ltd.** All rights reserved. No part of this publication or the Septel ISA (PCCS6) / PCCS3 hardware and the associated software may be reproduced, stored in a retrieval system or transmitted in any form or by any means without the prior written permission of DataKinetics Ltd.

Septel is a trademark of DataKinetics Ltd
SCbus is a trademark of the Dialogic Corporation.
MVIP is a trademark of Natural MicroSystems Corporation.
SCO is a registered trademark of The Santa Cruz Operation, Inc.
UNIX is a registered trademark of UNIX System Laroratories, Inc.
QNX is a registered trademark of QNX Software Systems Ltd.
Windows and **Windows NT** are registered trade marks of Microsoft Corporation.

Document reference: U08SSP

Revision History

Issue	Date	Changes
1	27-Feb-97	
2	20-Mar-97	Single manual for UNIX, QNX and Windows NT. SCO UNIX and QNX packages support both PCCS3 and PCCS6 boards, device driver installation and options have changed. Applicable to the following Development Package releases: SCO UNIX V3.00, QNX V3.00 Windows NT V1.00.
3	24-Mar-01	Manual renamed (Previously called "PCCS3 / PCCS6 SS7 Programmer's Manual for UNIX, QNX and Windows NT"). It now covers all supported operating systems except DOS. Support for Linux added. class field in header structure changed to hclass. Additional information on interfacing to the SCbus and configuring Line Interface Units. Applicable to the following Development Package releases: Septel Dev Package for Windows NT – V1.01 Septel Dev Package for Linux – V1.02 Septel Dev Package for QNX – V1.01 PCCS Dev Package for SCO UNIX – V3.05

Contents

1. INTRODUCTION 7
    1.1 Related Documentation 7

2. INSTALLATION 8
    2.1 Introduction 8
    2.2 Software Installation for Windows NT 9
        2.2.1 Installing the Septel Development Package for Windows NT 9
        2.2.2 Selecting device driver settings 11
        2.2.3 Configuring the device driver 11
    2.3 Software Installation for Linux 12
        2.3.1 Installing the PCCS Development Package for Linux 12
        2.3.2 Selecting device driver settings 13
        2.3.3 Creating the device node 13
        2.3.4 Loading the device driver 14
        2.3.5 Verifying device driver loading 15
    2.4 Software Installation for SCO UNIX 17
        2.4.1 SCO UNIX Device Driver Installation Procedure 17
        2.4.2 SCO UNIX Device Driver Removal Procedure 20
        2.4.3 Adding and Removing Boards under SCO UNIX 21
        2.4.4 Development Package Installation Procedure 22
        2.4.5 Development Package Removal Procedure 23
        2.4.6 User Part Development Package Installation Procedure 24
    2.5 Software Installation for QNX 25
        2.5.1 Installing the Septel Development Package for QNX 25
        2.3.2 Selecting device driver settings 26
        2.3.3 Running the device driver 26

3. CONFIGURATION AND OPERATION 28
    3.1 Overview 28
        3.1.1 System Structure 28
    3.2 System Configuration 30
        3.2.1 System configuration file syntax 30
        3.2.2 Generating system.txt 32
    3.3 Protocol Configuration 34
        3.3.1 Protocol Configuration using the s7_mgt utility 34
        3.3.2 Protocol Configuration using individual messages 34
    3.4 Using the Septel ISA SCbus 36
        3.4.1 Introduction 36
        3.4.2 Switching Model 36
        3.4.3 Static Initialisation 37
        3.4.4 Dynamic Operation 38
        3.4.5 Example code - Building and sending SC_LISTEN 39

4. PROGRAM EXECUTION 40
    4.3 Program execution under Windows NT 41
    4.3 Program execution under SCO UNIX and Linux 42
    4.4 Program execution under QNX 43
    4.5 Developing a user application 44

5. APPLICATION MESSAGE SPECIFICATIONS 45
5.1 Overview 45
5.1.1 General Configuration Messages 45
5.1.2 Hardware Control Messages 45
5.1.3 MTP Interface Messages 46
5.1.4 Event Indication Messages 46
5.1.5 Message Type Table 46
5.2 General Configuration Messages 48
5.2.1 SSD Reset Request 48
5.2.2 Board Reset Request 50
5.2.3 Board Configuration Request 52
5.2.4 MTP Route Configuration Request 59
5.3 Hardware Control Messages 62
5.3.1 LIU Configuration Request 62
5.3.2 LIU Control Request 66
5.3.3 LIU Read Configuration Request 68
5.3.4 LIU Read Control Request 69
5.3.5 LIU SCbus Initialisation Request 70
5.3.6 SCbus Listen Request 72
5.3.7 SCbus Pattern Request 74
5.3.8 Reset Switch Request 76
5.3.9 Set Output Request 77
5.3.10 SCbus Connect Request 81
5.3.11 Configure Clock Request (PCCS3) 85
5.3.12 Configure Clock Request (Septel ISA PCCS6) 87
5.3.13 Configure Carrier Request 90
5.4 MTP Interface Messages 92
5.4.1 MTP Link Activation Request 92
5.4.2 MTP Link Deactivation Request 93
5.4.3 MTP Transfer Request 94
5.4.4 MTP Transfer Indication 95
5.4.5 MTP Pause Indication 96
5.4.6 MTP Resume Indication 97
5.4.7 MTP Status Indication 98
5.5 Event Indication Messages 99
5.5.1 Board Status Indication 99
5.5.2 LIU Status Indication 100
5.5.3 Error Indication 101
5.5.4 MTP2 Level 2 State Indication 103
5.5.5 MTP2 Q.791 Event Indication 104
5.5.6 MTP3 Q.791 Event Indication 105

APPENDIX A: PROTOCOL CONFIGURATION COMMANDS 106
A1 Physical Interface Parameters 106
A1.1 PCCS3 Board Configuration 106
A1.2 Septel ISA (PCCS6) Board Configuration 107
A1.4 LIU_CONFIG Command 109
A1.5 LIU_SC_DRIVE Command 111
A1.6 SCBUS_LISTEN Command 113
A1.7 SCBUS_PATTERN Command 114
A2 MTP Parameters 115

A2.1 Global MTP configuration 115
A2.2 MTP Link Set 116
A2.3 MTP Signalling Link 117
A2.4 MTP Route 119
A2.5 MTP User Part 121
A3 ISUP Parameters 122
A3.1 Global ISUP Configuration 122
A3.2 ISUP Circuit Group Configuration 123
A4 TUP Parameters 125
A4.1 Global TUP Configuration 125
A4.2 TUP Circuit Group Configuration 126
A5 NUP Parameters 127
A5.1 Global NUP Configuration 127
A5.2 NUP Circuit Group Configuration 128

APPENDIX B: LIBRARY FUNCTION REFERENCE 129
B1.1 rpackbytes 129
B1.2 runpackbytes 130

1. INTRODUCTION

The DataKinetics Septel range of board level products includes specialised E1 / T1 signalling cards for use in ISA, PCI and Compact PCI systems. Two cards are available in an ISA form factor - the Septel ISA (formerly PCCS6) and the PCCS3. (Whilst still supported, the PCCS3 should be considered as obsolete for new designs).

In addition to E1/T1 interfaces, both ISA boards contain an embedded signalling processor to handle the SS7 signalling protocol and support various industry standard inter-card PCM highways (MVIP, SCbus and PEB).

As a hardware platform the Septel ISA card provides an ideal platform for running the DataKinetics System7 software modules for the realisation of Signalling System Number 7 protocol stacks. It may be used under any of the following operating systems: Windows NT, Linux, SCO UNIX and QNX.

This manual is the Programmer's Manual for the Septel ISA and PCCS3 cards. It is targeted at systems developers who will be integrating the cards and developing applications that will make use of the underlying SS7 protocol stack. The manual includes information on software installation, system configuration, protocol configuration and operation of the board and SS7 software stack.

The manual should be used in conjunction with the appropriate User Manual for the board and the Programmer's Manuals for the individual protocol modules as detailed below.

Users of the Septel PCI, Septel cP and Septel SIU products should refer instead to the documentation for those products.

1.1 Related Documentation

DataKinetics Septel ISA (PCCS6) User Manual

DataKinetics PCCS3 User Manual

DataKinetics System7 - ISUP Programmer's Manual

DataKinetics System7 - TUP Programmer's Manual

DataKinetics System7 - NUP Programmer's Manual

DataKinetics System7 - SCCP Programmer's Manual

DataKinetics System7 - TCAP Programmer's Manual

DataKinetics System7 - Software Environment Programmer's Manual

2. INSTALLATION

2.1 Introduction

Prior to commencing software installation the hardware should have been configured as described in the Septel ISA (PCCS6) User Manual or the PCCS3 User Manual.

This programmer's manual covers the installation and use of the software contained on the following distribution disks:

Septel Development Package for Windows NT
Septel Development Package for Linux
PCCS Device Driver for SCO UNIX
PCCS Development Package for SCO UNIX
Septel Development Package for QNX
System7 - User Part Development Package
System7 - Binary for PCCS3 (various protocols)
System7 - Binary for PCCS6 (various protocols)

The **Development Package** contains the device driver, header files and library functions for use by an application, a number of executables to be run as part of the System7 environment, and a utility program to configure the protocol software. (The Device Driver for SCO UNIX is shipped on a separate disk to the Development Package as the installation procedure involves rebuilding the kernel). The installation of this package is described in the following sections.

The **User Part Development Package** contains example source code to illustrate the techniques used for interfacing with the System7 software modules. It is distributed on a DOS format disk and is applicable to all supported operating systems. Copy the contents of the User Part Package distribution disk onto the development machine maintaining the sub-directory structure.

The **System7 Binary** disks are DOS format disks containing the operating software for either the Septel ISA or the PCCS3. This is in the form of a single binary file ('the code file') which is downloaded to the board at run-time by the driver program. Code files for PCCS3 have a file suffix .dc1 whilst code files for the Septel ISA (PCCS6) have a file suffix of .dc2. The title of the disk depends on which protocols (if any) in addition to the SS7 Message Transfer Part (MTP) (which is always included) are supported by the code file. Examples of disk titles are listed below:

System7 Binary for PCCS3 - MTP	(MTP for PCCS3)
System7 Binary for PCCS3 - ISUP	(MTP & ISUP for PCCS3)
System7 Binary for PCCS6 - TUP	(MTP & TUP for PCCS6)
System7 Binary for PCCS6 - TCAP	(MTP, SCCP & TCAP for PCCS6)

The code file should be copied onto the target machine maintaining binary file integrity, it will subsequently be downloaded onto the board at run time.

2.2 Software Installation for Windows NT

2.2.1 Installing the Septel Development Package for Windows NT

The Septel Development Package for Windows NT is distributed either on a DOS format disk or electronically by email or from the DataKinetics web site. In the latter cases the distribution is in the form of a single zip file (septel.zip) which should be unzipped and placed on a clean DOS format disk to replicate the normal distribution and serve as a backup.

The installation procedure involves running the "Install Shield" procedure from the distribution disk. First close all other applications then insert the disk into the floppy disk drive of the target machine. You must be logged on as a user with Administrator privileges to install the software.

If the target machine does not automatically detect the installation disk then run the program 'Setup.exe' from the distribution disk. The licence agreement must be read and accepted before installation can proceed.

The installation procedure prompts for an installation directory. The default directory is c:\septel. If required, the default directory can be modified.

The following files (or similar) are transferred to the installation directory:

pccsxdvr.sys	The Windows NT device driver for the Septel ISA
pccsxcfg.exe	Board installation utility
gctlib.lib	Library to be linked with user's application (Microsoft)
gctlibb.lib	Library to be linked with user's application (Borland)
INC	Sub-directory containing header files for use with user's application
system.txt	Example system configuration file
config.txt	Example protocol configuration file
gctload.exe ssd.exe ssd_poll.exe tick_nt.exe tim_nt.exe s7_mgt.exe s7_log.exe s7_play.exe mtpsl.exe upe.exe	Executables for use as described elsewhere in this manual
gctserv.exe servcfg.exe	Utilities for configuring gctload.exe to run as a Windows NT service
rsi.exe rsi_lnk.exe rsicmd.exe hstmgr.exe ssds.exe	Files not used with the Septel ISA or PCCS3
uninst.isu	File used by Windows NT if uninstalling the development package

The device driver must be copied from the installation directory into the SYSTEM32 directory on your target machine as follows:

```
copy pccsxdvr.sys c:\winnt\system32
```

If Windows NT is installed in a directory other than “c:\winnt” on your system then substitute the correct directory name in the above command. Configuring the device driver for use in your system is described in a later section of this programmer’s manual.

Note that the Development Package also contains files for use by customers using other Septel products which are not relevant to Septel ISA operation.

Installation is now complete. It is recommended that the user does not modify the files in the installation directory but instead creates a working directory into which all the necessary files are copied.

Before installing a new version of the Septel Development Package for NT it is necessary to remove any previous release from the system.

Firstly, the pccsxdvr.sys file should be deleted from the Windows NT system32 directory:

```
del c:\winnt\system32\pccsxdvr.sys
```

Use the Windows NT uninstall feature to remove the other files from the system:

1. Open Control Panel by selecting Settings: Control Panel from the Start Menu.
2. Click “Add/Remove Programs” and then select “Septel Development Package” from the list in the Install/Uninstall page.
3. Click the “Add/Remove…” button.
4. Press the “Yes” button on the pop-up dialog box to confirm the file removal.
5. Restart the computer before installing the new package.

2.2.2 Selecting device driver settings

Prior to configuring the device driver, the following must be established:

- The number of boards to be installed

- A single, unused, contiguous, 4k byte region of memory (this is used by all Septel ISA boards).

- An unused, contiguous, 4 byte region of I/O for each Septel ISA board.

The Windows NT utility, WINMSD.EXE can be used to find suitable free I/O addresses and a block of free memory.

Having decided upon suitable settings, the address switch and interrupt jumper on each Septel ISA card should be set up in accordance with the instructions in the **Septel ISA User Manual**.

2.2.3 Configuring the device driver

The Windows NT device driver is configured using the executable pccsxdvr.exe.

This can only be run by users with accounts belonging to the “Administrators’ user group.

The syntax is:

```
pccsxcfg  [-m -p -n -b] <add | remove | modify> <device_driver>

where :
      -n    Number of boards to install.
      -p    base I/O port address.
      -m    base memory address.
      -b    board_id. (0, 1, 2, etc)
```

For example, to install two Septel ISA boards (board_id=0 and board_id=1) at I/O addresses 0x300 and 0x304 respectively and memory addresses 0xd0000 and 0xd1000 respectively the following command would be used:

```
pccsxcfg -n2 -p0x300 -m0xd0000 add c:\winnt\system32\pccsxdvr.sys
```

To remove the device driver and any boards installed use:

```
pccsxcfg remove
```

To change either the I/O address or the memory address for an existing board:

```
pccsxcfg -b0 -p0x38000 -m0xe0000 modify c:\winnt\system32\pccsxdvr.sys
```

2.3 Software Installation for Linux

2.3.1 Installing the PCCS Development Package for Linux

The PCCS Development Package for Linux is distributed either on a DOS format disk or electronically by email or from the DataKinetics web site. The distribution is in the form of a single compressed file called linux.z

The file should be transferred to the Linux development system taking care to ensure that the '.z' file suffix is in lower case. It should then be uncompressed and extracted using the commands shown below:

```
gzip -d linux
tar -xvf lnx
```

The files are extracted into the current working directory, it is recommended that you create a new directory to serve as the root directory for the System7 software. The following files (or similar) are contained on the distribution:

pccs22x.o pccs20x.o	Linux device drivers for the Septel ISA card. (two drivers are supplied, one for use with V2.0.x kernels and the other for V2.2.x kernels)
gctlib.lib	Library to be linked with user's application
INC	Sub-directory containing header files for use with user's application
system.txt	Example system configuration file
config.txt	Example protocol configuration file
gctload ssd tick_lnx tim_lnx s7_mgt s7_log s7_play mtpsl upe	Executables for use as described elsewhere in this manual
hstmgr rsi rsi_lnk rsicmd sptpci.o ssds	Files not used with the Septel ISA or PCCS3

2.3.2 Selecting device driver settings

The Linux device driver is a run-time loadable module. It can be loaded manually using the 'insmod' command and does not need the kernel to be rebuilt.

Before the module can be loaded, the following must be established:

- An unused major character device number.

- The number of boards to be installed

- A single, unused, contiguous, 4-Kilobyte region of memory (this is used by all Septel ISA boards).

- An unused, contiguous, 4 byte region of I/O for each Septel ISA board.

- An unused interrupt number.

Having decided upon suitable settings, the address switch and interrupt jumper on each Septel ISA card should be set up in accordance with the instructions in the **Septel ISA User Manual**.

2.3.3 Creating the device node

A device file must be created for each Septel ISA card. The device file must be assigned an unallocated, major character device number and a minor device number. First it is necessary to establish an unallocated major device number. If your Linux distribution supports the /proc file system, the major device numbers currently in use by character and block devices can be displayed using:

```
more /proc/devices
```

Otherwise refer to the operating system documentation for details on how to allocate a major character device number.

Once a major device number is known, the 'mknod' command must be run once for each board in the system and the attributes of the device file must set using 'chmod'. Both these operations should be performed with root priviledges.

The appropriate syntax is:

```
cd /dev
mknod pccs<MINNO> c <MAJNO> <MINNO>
chmod 0666 pccs<MINNO>
```

where: MAJNO is the major device number, MINNO is in the range 0 to 15. MINNO is the board_id.

For example, the commands to create 2 Septel ISA boards using major character device number 126 are:

```
cd /dev
mknod pccs0 c 126 0
mknod pccs1 c 126 1
chmod 0666 pccs*
```

2.3.4 Loading the device driver

The Linux Development Package contains two device driver modules: pccs20x.o and pccs22x.o for use with different Linux kernels V2.0.x and V2.2.x, respectively. The kernel version can be determined using the 'uname' command as follows:

```
uname -a
```

This will generate output similar to the following including the kernel version as the third field (the minor kernel is not significant):

```
Linux linuxpci 2.2.12-20 #1 Mon Sep 27 10:25:54 EDT 1999 i686 unknown
```

The device driver should be manually loaded using the 'insmod' command from the directory containing the device driver (eg pccs22x.o) file whilst logged on with root privileges. The command can be executed in either a system start-up script or by the user from a command shell if required.

The syntax is:

```
insmod -f pccs20x.o majno= irq= btype= ioaddr= memaddr=
```

or

```
insmod -f pccs22x.o majno= irq= btype= ioaddr= memaddr=
```

depending on whether the driver for the V2.0.x or V2.2.x Linux kernel is required.

The command line options include global options (majno and irq) that are common to all boards in the system and per-board options (btype, ioaddr and memaddr) that need to contain a separate value for each board installed. The -f option is used to force loading of the driver even if it was built using a different kernel version.

Options values can be entered as hex or decimal numbers, hex values use a 0x prefix (e.g. 10 can specified as 10 or 0x0a).

For example (using 2 Septel ISA boards):

```
insmod –f pccs22x.o irq=0x7 btype=6,6 ioaddr=0x200,0x204
memaddr=0xd0000,0xd0000 majno=40
```

<majno> - Major device number

The 'majno' option is used to specify the major device number.

<irq> - Interrupt number

The ’irq’ option is used to specify the interrupt number. All Septel ISA boards in the system use the same interrupt number. The hardware should be correctly configured to use the interrupt number specified in the command line.

The board requires exclusive use of the interrupt. If the interrupt is already in use, the device driver will not load. If the target system supports the ’/proc’ file system, any interrupts currently in use can be displayed using:

```
more /proc/interrupts
```

<btype> - Board type

The ’btype’ option is used to specify a list of board types for each board installed. It should be set to 6 for Septel ISA (PCCS6) boards and 3 for PCCS3 boards.

If multiple boards are in use then the option must contain a list of values as shown below assuming a system with 4 Septel ISA boards:

```
btype=6,6,6,6
```

<ioaddr> - I/O address

The ’ioaddr’ option is used to configure a unique base I/O address for each Septel ISA board. The board will use a block of 4 consecutive I/O addresses commencing at the base address. If the target system supports the ’/proc’ file system, the IO ports currently used can be shown using:

```
more /proc/ioports
```

If multiple boards are in use then the option must contain a list of values as shown below assuming a system with 4 Septel ISA boards:

```
ioaddr=0x200,0x204,0x300,0x304
```

<memaddr> - Memory address

The ’memaddr’ option is used to configure the base memory address for each Septel ISA board. The board uses a block of 4k byte consecutive memory addresses. Note that each board can use the same block of memory.

If multiple boards are in use then the option must contain a list of values as shown below assuming a system with 4 Septel ISA boards:

```
memaddr=0xd0000,0xd0000,0xd0000,0xd0000
```

2.3.5 Verifying device driver loading

When the device driver is loaded it will output status messages to the system log. The system log can be displayed using the following command:

```
dmesg | more
```

An example of messages written to the system log by the driver is:

```
PCCS Device Driver for Linux 2.2.x V2.01.
Copyright (C) 1999-2000 DataKinetics Ltd. All rights reserved.
Using: IRQ 0x7, Major device number 80
Configured 2 PCCS board(s):
pccs[0]: PCCS6 @ IO 0x200 DPM 0xd0000
pccs[1]: PCCS6 @ IO 0x204 DPM 0xd0000
```

2.4 Software Installation for SCO UNIX

The PCCS Device Driver for SCO UNIX and the PCCS Development Package for SCO UNIX are both installed and removed using the SCO UNIX **custom** utility. A description of **custom** can be found in the *SCO UNIX Operating System Installation Guide* in the appendix entitled "Installing and removing additional software".

If you are running Open Desktop the utility is described in the *SCO Open Desktop Installation and Update Guide* in the appendix entitled "Installing and removing additional software".

The use of the custom utility provides a controlled mechanism for installing or removing individual packages as complete entities.

2.4.1 SCO UNIX Device Driver Installation Procedure

If you are updating an existing installation of the DataKinetics PCCS Device Driver for SCO UNIX is first necessary to remove the existing package (refer to the Device Driver Removal Procedure).

The installation and removal of the PCCS Device Driver package will modify and rebuild the UNIX kernel. It is therefore necessary to ensure that all users are logged off the system before installing or removing the package.

The installation procedure is as follows:

1. If you are not already in System Maintenance mode, log out and log in as *root*. Shutdown the machine using the command:

   ```
   /etc/shutdown -i1
   ```

   *Note : **shutdown -i1** brings the system safely into the System Maintenance mode. Further information on the **shutdown** command can be found on the **shutdown** (ADM) manual page.*

2. Enter the *root* password at the prompt, as shown below:

   ```
   INIT: New run level: S

   INIT: SINGLE USER MODE

   Type CONTROL-d to proceed with normal start-up
   (or give root password for system maintenance):
   ```

3. Access System Maintenance (single-user) mode by typing the *root* password. You now have access to all the system files, so you should be careful not to overwrite, delete or corrupt any files accidentally.

4. At the prompt, type `custom` and press <Enter>

5. When the Custom menu appears, Select the Install option and press <Enter> to start the installation. **custom** displays the Install screen, listing currently-installed software and prompting you to select a product to install

6. Select the option "A New Product" and press <Enter>

7. **custom** offers the choice of installing the Entire Product, Disks or Files. Select the option "Entire Product" and press <Enter>.

8. **custom** prompts you to insert Distribution Floppy Volume 1.

9. Insert the PCCS Device Driver for SCO UNIX disk into the drive, and press <Enter>. A confirmation message will appear at the top of the screen as shown below:

   ```
   Installing custom data files....
   ```

   Leaving the Install screen, **custom** then checks for the correct installation environment. During this procedure, the following message will appear briefly:

   ```
   Executing Product Prep script
   ```

   After the prep script, **custom** will prompt you to insert the DataKinetics PCCS Device Driver Package Floppy Volume 1, and the Continue option will be highlighted. This is the disk which you have already placed in the drive.

10. Check that the disk is in the drive, and then press <Enter>. The following message will appear at the top of the screen:

    ```
    Extracting files.....

    Executing DataKinetics PCCS Device Driver Package Init Script
    ```

11. You will be prompted in turn for the type, address and interrupt number of each board you are installing. The default values will be shown in brackets - if the default value is acceptable simply press return. After you have set the parameters for a board you will be asked whether you wish to install another board. Note that the default address parameter will change depending on the number of the board being installed, so you can accept the default address for each board unless this clashes with other devices in your PC.

    Note : The same interrupt number should be selected for all boards. This interrupt number must be the same as that selected using the option links on the board as described in the User Manual for the board(s) you are installing.

    Each board installed is automatically allocated a sequential board id, starting from 0. This board id is subsequently used to specify which board a message is to be sent to or was received from.

    When you have finished installing boards, answer no to the prompt asking if you wish to install another board.

12. The following message will appear:

```
Configuring for kernel driver installation...

The kernel link kit has been modified.
You must relink the kernel to receive the new functionality.
Do you want to relink the kernel at this time ? (y/n)
```

Enter y and press <Enter> to relink the kernel. During this process the following message will be displayed:

```
The UNIX Operating System will now be rebuilt.
This will take a few minutes.  Please wait.
```

13. When the kernel has been rebuilt, the following message will be displayed:

```
The UNIX Kernel has been rebuilt.

Do you want this kernel to boot by default? (y/n)
```

14. Enter y and press <Enter>. The following message will appear:

```
Do you want the kernel environment rebuilt? (y/n)
```

15. Enter y and press <Enter>. The following message will appear:

```
The kernel has been successfully linked and installed
To activate it, reboot your system.

Setting up new kernel environment
```

16. When the new environment has been set up, the following message will appear:

```
Press any key to continue
```

17. Press any key to continue the installation process.

    The following message appears briefly on the screen:

```
Checking file permissions....
```

    You are now returned to the main **custom** menu. The DataKinetics PCCS Device Driver package will have been added to the list of installed software.

19. To quit **custom**, use the cursor to highlight the Quit option, and press <Enter>. You are prompted to confirm that you want to quit, with the Yes option highlighted. Press <Enter>.

2.4.2 SCO UNIX Device Driver Removal Procedure

The Device Driver Removal Procedure is used before installing a new version of the package to first remove the existing package. The procedure is as follows:

1. Log in to the system as *root* in System Maintenance Mode, as described under Installation above.

2. At the prompt, type `custom` and press <Enter>

3. Select Remove, and press <Enter>.

4. A list of installed software will appear. Using the cursor, select the DataKinetics PCCS Device Driver package and press <Enter>.

5. The packages list will appear. Select All, and press <Enter>.

6. A message will appear requesting confirmation of removal. Confirm the removal of the PCCS Device Driver package.

7. **custom** will remove the PCCS Device Driver package.

8. The following message will appear:

```
Configuring for kernel driver de-installation...

The kernel link kit has been modified.
You must relink the kernel to receive the new functionality.
Do you want to relink the kernel at this time ? (y/n)
```

   If you are updating the PCCS Device Driver package enter n and press <Enter>. If you are removing it enter y and press <Enter>. If you chose to rebuild the kernel, proceed as described in the Installation procedure above from the step describing the kernel rebuilding.

9. When the following message appears:

```
Press any key to continue
```

   press any key to continue the installation process. The following message appears briefly on the screen:

```
Checking file permissions....
```

   You are now returned to the main **custom** menu. The DataKinetics PCCS Device Driver package will have been removed from the list on your screen.

   To install a new version of the driver, follow the installation procedure described above from the step after running **custom**. To exit **custom**, select Quit, and press <Enter>. Then select the Yes option, and press <Enter> to quit.

2.4.3 Adding and Removing Boards under SCO UNIX

Boards may be added to or removed from the system after the initial PCCS Device Driver installation.

Note : After adding or removing boards the UNIX kernel must be re-linked.

1. To add a board to the system, enter:

   ```
   /usr/lib/dkcs/addboard
   ```

   Follow the prompts to set the board options as described under “Installation procedure” above. The board is allocated the next sequential board id. Now follow the instructions under “Relinking the UNIX kernel”.

2. To remove a board from the system, enter:

   ```
   /usr/lib/dkcs/rmvboard
   ```

   The board which was last installed will be removed from the system (ie. the board with the highest board id). Now follow the instructions under “Relinking the UNIX kernel”.

3 To relink the UNIX kernel enter:

   ```
   /etc/conf/cf.d/link_unix
   ```

   During this process the following message will be displayed:

   ```
   The UNIX Operating System will now be rebuilt.
   This will take a few minutes.  Please wait.
   ```

   When the kernel has been rebuilt, the following message will be displayed :

   ```
   The UNIX Kernel has been rebuilt.

   Do you want this kernel to boot by default? (y/n)
   ```

   Enter y and press <Enter>. The following message will appear:

   ```
   Do you want the kernel environment rebuilt? (y/n)
   ```

   Enter y and press <Enter>. The following message will appear:

   ```
   The kernel has been successfully linked and installed.
   To activate it, reboot your system.
   ```

2.4.4 Development Package Installation Procedure

If you are updating an existing installation of the PCCS Development Package for SCO UNIX it is first necessary to remove the existing package (refer to the Development Package Removal Procedure).

The installation procedure is as follows :

1. If you are not already in System Maintenance mode, log off and log in as *root*.

2. At the prompt, type `custom` and press <Enter>

3 When the **custom** menu appears, the Install option will be highlighted. Press <Enter> to start the installation. **custom** displays the Install screen, listing currently-installed software and prompting you to select a product to install

4. Select the option "A New Product" and press <Enter>

5. **custom** offers the choice of installing the Entire Product, Disks or Files.

6 Select the option "Entire Product" and press <Enter>.

7. **custom** prompts you to insert Distribution Floppy Volume 1.

8. Insert the PCCS Development Package for SCO UNIX disk into the drive, and press <Enter>.

   A confirmation message will appear at the top of the screen as shown below:

   ```
   Installing custom data files....
   ```

   Leaving the Install screen, **custom** then checks for the correct installation environment. During this procedure, the following message will appear briefly:

   ```
   Executing Product Prep script
   ```

9. **custom** will prompt you to insert the DataKinetics PCCS Development Package Floppy Volume 1, and the Continue option will be highlighted. This is the disk which you have already placed in the drive. Check that the disk is in the drive, and press <Enter>. The following message will appear at the top of the screen:

   ```
   Extracting files.....

   Executing DataKinetics PCCS Development Package Init Script
   ```

10. The following message will appear:

    ```
    Press any key to continue
    ```

11. Press any key to continue the installation process. The following message appears briefly on the screen:

    ```
    Checking file permissions....
    ```

12 You are now returned to the main **custom** menu. The DataKinetics PCCS Development package will have been added to the list of installed software. To quit **custom**, use the cursor to highlight the Quit option, and press <Enter>. You are prompted to confirm that you want to quit, with the Yes option highlighted. Press <Enter>.

13 Once installed the Development Package files are stored under the following directories:

/usr/lib/dlkdev/BIN	Executables
/usr/lib/dkldev/EXE	Configuration and download files
/usr/lib/dkldev/LIB	Library files
/usr/lib/dkldev/SRC	Source files

The use of the files is described in later sections.

2.4.5 Development Package Removal Procedure

The Development Package Removal Procedure is used before installing a new version of the package to first remove the existing package. The procedure is as follows:

1. Log in to the system as *root* in System Maintenance Mode, as described under Installation above.

2. At the prompt, type `custom` and press <Enter>

3. Select Remove, and press <Enter>.

4. A list of installed software will appear. Select the DataKinetics PCCS Development package and press <Enter>.

5. The packages list will appear. Select All, and press <Enter>.

6. A message will appear requesting confirmation of removal. Confirm the removal of the PCCS Development package.

7. **custom** will remove the PCCS Development package.

8. You are now returned to the main **custom** menu. The DataKinetics PCCS Development package will have been removed the list on your screen.

   To exit **custom**, select Quit, and press <Enter>. Then select the Yes option, and press <Enter> to quit.

2.4.6 User Part Development Package Installation Procedure

The User Part Development Package is supplied on a DOS format disk (the package is common to other operating systems). Installation involves copying the files from the disk onto the target system.

If you have previously installed a version of the User Part Development Package prior to V2.00 it is first necessary to remove the existing package using the custom utility as described here:

1. Log in to the system as *root* in System Maintenance Mode, as described under Installation above. At the prompt, type `custom` and press <Enter>

2. Select Remove, and press <Enter>. A list of installed software will appear. Select the User Part Development package and press <Enter>.

3. The packages list will appear. Select All, and press <Enter>. A message will appear requesting confirmation of removal. Confirm the removal of the User Part Development package. **custom** will remove the User Part Development package.

4. You are now returned to the main **custom** menu. The User Part Development package will have been removed the list on your screen. To exit **custom**, select Quit, and press <Enter>. Then select the Yes option, and press <Enter> to quit.

2.5 Software Installation for QNX

2.5.1 Installing the Septel Development Package for QNX

The Septel Development Package for QNX is distributed either on a DOS format disk or electronically by email or from the DataKinetics web site. The distribution is in the form of a single compressed file called QNX.F.

The file should be transferred to the QNX development system taking care to ensure that the '.F' file suffix is in upper case. It should then be uncompressed and extracted using the commands shown below:

```
freeze -d qnx.F
tar -xvf qnx
```

The files are extracted into the current working directory, it is recommended that you create a new directory to serve as the root directory for the System7 software. The following files (or similar) are contained on the distribution:

pccs.dev	QNX device driver for the Septel ISA card.
gctlib.lib	Library to be linked with user's application
INC	Sub-directory containing header files for use with user's application
system.txt	Example system configuration file
config.txt	Example protocol configuration file
gctload qbuf sema ssd tick_qnx tim_qnx s7_mgt s7_log s7_play mtpsl upe	Executables for use as described elsewhere in this manual
hstmgr rsi rsi_lnk rsicmd sptpci.dev ssds	Files not used with the Septel ISA or PCCS3

2.3.2 Selecting device driver settings

Priot to running the device driver the following must be established:

- The number of boards to be installed

- A single, unused, contiguous, 4k byte region of memory (this is used by all Septel ISA boards).

- An unused, contiguous, 4 byte region of I/O for each Septel ISA board.

- An unused interrupt number.

Having decided upon suitable settings, the address switch and interrupt jumper on each Septel ISA card should be set up in accordance with the instructions in the **Septel ISA User Manual**.

2.3.3 Running the device driver

Prior to starting the protocol software it is necessary to run the Device Driver. Usually this is achieved by adding an entry in the sysinit.1 file so that every time the machine starts up the driver is installed automatically. Command line parameters are used to specify the number of boards, the base address (in memory and I/O space) and the interrupt number.

The syntax is:

```
pccs.dev -T<type> -I<irq> -n<num_boards> -B<base_address> -M<base_IO>
```

A typical command line to run the device driver with 8 PCCS6 boards is:

```
pccs.dev -T6 -I15 -n8 -B0xe0000 -M0x200
```

A typical command line to run the device driver with 4 PCCS3 boards is:

```
pccs.dev -T3 -I15 -n4 -B0xe0000
```

The parameters have the following meanings:

-n Number of boards

The -I option specifies the maximum number of boards to be supported by the device driver. (This must not exceed 16).

-I Interrupt number

The -I option is used to specify the interrupt number. All Septel ISA boards in the system use the same interrupt number. The hardware should be correctly configured to use the interrupt number specified in the command line.

-T Board type

The -T option is used to specify the board type. It should be set to 6 for Septel ISA (PCCS6) boards and 3 for PCCS3 boards.

-M I/O address

The -M option is used to configure a base I/O address for the first Septel ISA board. The board uses a block of 4 consecutive I/O addresses commencing at the value specified. Subsequent boards are mapped at successive 4 byte I/O addresses. This option is not required for PCCS3 boards.

-B Base memory address

The -B option is used to configure the base memory address for each Septel ISA board. The board uses a block of 4k byte consecutive memory addresses. Note that each Septel ISA (PCCS6) board uses the same block of memory whilst PCCS3 boards are mapped at successive 4k blocks

3. CONFIGURATION AND OPERATION

3.1 Overview

Prior to attempting software configuration the user should gain an appreciation of the flexibility of the protocol stack, what run-time options exist and which mechanisms are used to select particular features. This section gives an overview of these options. The user should also read the *System7 - Software Environment Programmer's Manual* which describes the basic principles of modules and message passing.

3.1.1 System Structure

The SS7 software running on the board communicates with an application that runs on the main CPU of the host computer. The physical interface to the board uses dual port memory on the PCCS3 and a combination of dual port memory and 4 I/O locations on the Septel ISA (PCCS6). The dual port memory appears in the memory map of the host computer. The interface to the board is directly handled by a device driver and all message passing to and from the board is managed by a process (ssd) which runs on the host computer.

In addition to running the application on the host processor the user may, depending on the size of the overall system and the network topology, elect to run some of the SS7 protocol modules also on the host. In such cases the interface between the application and the SS7 protocol software remains identical. This allows for easy migration from a small system contained on a single board to a large system distributed over many boards with minimal changes to the application. The table below illustrates some possible practical system configurations for a telephony system.

	Small System	**Medium System**	**Large System**
Software running on Septel ISA board	MTP2 MTP3 ISUP / TUP / NUP	MTP2 MTP3	MTP2
Software running on Host CPU	User Application	ISUP / TUP / NUP User Application	MTP3 ISUP / TUP / NUP User Application
Number of boards	Single	Single board with signalling (although additional boards may be used to support voice only)	Multiple
Description	Suitable for small systems, supporting 1, 2 or 3 signalling links and up to 1024 voice circuits.	Suitable for systems with up to 3 signalling links where more than 1024 voice circuits are required.	Suitable for systems with more than 3 signalling links, and systems which require distribution of MTP2 over multiple boards.

The following abbreviations are used in the table:

MTP2 - Message Transfer Part - Level 2
MTP3 - Message Transfer Part - Level 3
ISUP - ISDN User Part
TUP - Telephony User Part
NUP - National User Part (UK)

In all cases the process called **ssd** (**S**ystem7 **S**oftware **D**river) is required to be run on the Host machine. This handles message transfer between the host and the board using the device driver. Note that under Windows NT and additional process **ssd_poll** must also be used to ensure the correct operation of ssd.

The selection of which protocol modules to run on the host is made by editing a text file **system.txt**. The user then runs the program **gctload** which reads the system configuration parameters from the file system.txt and starts up the selected processes bringing the system into operation.

For further details of the operation of **gctload** refer to the *System7 - Software Environment Programmer's Manual*

The following processes for use on the host are included in the distribution. All must be run on the host with the exception of **s7_mgt** which is optional:

gctload	Process to initialise the system environment and start up all other System7 processes running on the host, deriving the configuration from a text file (system.txt).
ssd	Process to interface with the device driver for passing messages to and from the board(s) and for downloading software to the board(s). (In the case of the SCO UNIX version this process also handles the tick and tim functions described below).
tick_nt **tick_lnx** **tick_qnx**	Protocol timer process to send periodic 'tick' notification to the tim_nt / tim_lnx / tim_qnx process which in turn handles protocol timers.
tim_nt **tim_lnx** **tim_qnx**	Process to receive periodic tick notification from tick_nt / tick_lnx / tick_qnx and handle protocol timers for all other processes.
s7_mgt	Process to perform single shot protocol configuration for all protocol modules, deriving the configuration parameters from a text file (config.txt). This process is optional. As an alternative to using it the user may elect to perform protocol configuration by sending messages directly to the other modules in the system.
s7_log	Utility process to allow messages received from the protocol stack to be logged to a text file. This is useful for diagnostic purposes when getting started.
s7_play	Utility process used to generate messages from a text file and send them into the system. This is useful for diagnostic purposes when getting started.
ssd_poll (NT only)	Process used in conjunction with ssd when running Windows NT only.
sema **qbuf** (QNX only)	Processes used only on QNX systems, both started automatically by gctload and used as part of the inter-process communication mechanism.

3.2 System Configuration

System configuration is handled by the program **gctload** which reads the system configuration data from a file called **system.txt.**

System initialisation requires first that a pool of message buffers are created for subsequent inter-process communication. Secondly that a message queue is created for each process that will run and that any message re-direction for modules that are running remotely is initialised. Then all processes can be started.

The program **gctload** exists to handle this initialisation sequence and create the inter-process communication environment. It reads input from a text file called system.txt, carries out all system initialisation and starts up all processes. system.txt is a user configurable file containing details of all the module identifiers known to the system, details of whether they are local modules or remote modules accessed by a local module (message redirection) and lists the command line for all processes to be started by gctload.

gctload creates a message queue for each of the local module identifiers. It subsequently expects a process to service it's message queue otherwise messages written to that queue will never be read causing eventual loss of system messages.

It initialises the message queue look-up table so that messages destined for modules that do not exist locally are re-directed to a message queue for a module that does exist locally.

Having created the system environment, gctload proceeds to spawn all processes listed in the system configuration file in the order listed.

Prior to running gctload the system configuration file **must** be edited to reflect the requirements of your system.

3.2.1 System configuration file syntax

The system configuration file is a text file used by gctload to configure the software environment.

The file syntax permits the use of comments to improve the readability of the file. Comments are inserted into the file by using an asterisk *; all characters on the line after the asterisk are ignored.

Numbers can be entered in either decimal or hexadecimal format. Hexadecimal numbers should be prefixed with 0x. For example the value eighteen can be entered in either of the following formats:

```
0x12        *(Hexadecimal)
18          *(Decimal)
```

The System Configuration File contains the following commands:

a) LOCAL commands to allow gctload to generate message queues for modules running locally.

b) REDIRECT commands to cause messages generated for modules not running locally to be redirected via a module that is running locally.

c) FORK_PROCESS commands advising gctload of any processes that need to be started locally.

The full syntax of each command is listed in the System7 Software Environment Programmer's Manual.

An example system.txt file is shown below:

```
*
* Example system.txt for the DataKinetics Windows NT Development
* Package.
*
* If neccessary, edit this file to reflect your configuration.
*
* Essential modules running on host:
*
LOCAL       0x20        * ssd - Board interface task
LOCAL       0x21        * ssd_poll – required by ssd
LOCAL       0x00        * tim_nt - Timer task
*
* Optional modules running on the host:
*
LOCAL       0xcf        * s7_mgt - Management/config task
LOCAL       0x2d        * upe - Example user part task
*
* Modules running on the board (all redirected via ssd):
*
* REDIRECT  0x23  0x20  * ISUP module
* REDIRECT  0x4a  0x20  * TUP module
* REDIRECT  0x4a  0x20  * NUP module
* REDIRECT  0x33  0x20  * SCCP module
* REDIRECT  0x14  0x20  * TCAP module
REDIRECT    0x22  0x20  * MTP3 module
REDIRECT    0x71  0x20  * MTP2 module
REDIRECT    0x10  0x20  * MVIP/SCbus/Clocking control module
REDIRECT    0x8e  0x20  * On-board management module
*
* Redirection of status indications:
*
REDIRECT    0xdf  0x2d  * LIU/MTP2 status messages -> upe
REDIRECT    0xef  0x2d  * Other indications -> upe
*
* Now start-up all local tasks:
*
FORK_PROCESS      ssd_poll.exe –d10
FORK_PROCESS      ssd.exe
FORK_PROCESS      tim_nt.exe
FORK_PROCESS      tick_nt.exe
FORK_PROCESS      s7_mgt.exe
FORK_PROCESS      upe.exe
*
```

3.2.2 Generating system.txt

This section describes the procedure for generating a system configuration file (system.txt) and details any operating specific differences in behaviour of the development packages.

First the file must contain LOCAL declarations for all modules that are to run on the host computer. As a minimum this must include the SSD module and (except for SCO UNIX) the timer module. Hence the following declarations must exist:

```
LOCAL 0x20        * ssd - Board interface task
LOCAL 0x00        * tim_nt/tim_lnx/tim_qnx - Timer task
```

On NT systems the ssd_poll program must also run, and this uses module id 0x21.

LOCAL declarations are also required for any optional modules running on the host. Typically this will include s7_mgt and the user's own application module. It may also include and host based protocol modules and the s7_log utility. For example:

```
LOCAL 0xcf        * s7_mgt - Management/config task
LOCAL 0x2d        * upe - Example user part task
LOCAL 0x3d        * s7_log - Prints messages to screen/file
```

Once all the LOCAL declarations are in place, REDIRECT commands should be added for all modules that are running on the board so that any messages destined for these modules are transported via ssd (module_id = 0x20) and the device driver to reach the board.

The following REDIRECT commands are always required:

```
REDIRECT    0x71  0x20  * MTP2 module
REDIRECT    0x10  0x20  * MVIP/SCbus/Clocking control module
REDIRECT    0x8e  0x20  * On-board management module
```

In addition REDIRECT commands are required for all protocols running on the board. This will usually include MTP3 and one or more user parts. Examples of these commands are given below:

```
REDIRECT    0x23  0x20  * ISUP module
REDIRECT    0x4a  0x20  * TUP module
REDIRECT    0x4a  0x20  * NUP module
REDIRECT    0x33  0x20  * SCCP module
REDIRECT    0x14  0x20  * TCAP module
REDIRECT    0x22  0x20  * MTP3 module
```

Having ensured that all modules running on the board are accessible it is then necessary to ensure that any status indications issued from the board will successfully arrive at a module running on the host. (If this does not happen then the system will quickly run out of available messages for inter-process communication). Two module_id's (0xdf and 0xef) require redirection to a suitable process running on the host, initially these messages should be redirected to the s7_log utility which will print out a line for each message received. Ultimately the user's own application will expect to receive these notifications.

```
REDIRECT    0xdf  0x3d  * LIU/MTP2 status messages -> s7_log
REDIRECT    0xef  0x3d  * Other indications -> s7_log
```

It is next necessary to include FORK_PROCESS commands for all modules running on the host computer. All systems require ssd to be run (Windows NT users must also run the ssd_poll binary to ensure correct operation of the system), all except SCO UNIX users must also run the tick and tim binaries.

Therefore for Windows NT users, the following FORK_PROCESS commands are mandatory:

```
FORK_PROCESS      ssd.exe
FORK_PROCESS      ssd_poll -d10
FORK_PROCESS      tim_nt.exe
FORK_PROCESS      tick_nt.exe
```

For Linux, the following FORK_PROCESS commands are mandatory:

```
FORK_PROCESS      ssd
FORK_PROCESS      tim_lnx
FORK_PROCESS      tick_lnx
```

For SCO UNIX, the following FORK_PROCESS commands are mandatory:

```
FORK_PROCESS      ssd
```

For QNX, the following FORK_PROCESS commands are mandatory:

```
FORK_PROCESS      ssd
FORK_PROCESS      tim_qnx
FORK_PROCESS      tick_qnx
```

QNX users should note that gctload will automatically run the binaries sema and qbuf and that these should NOT be referred to in the system.txt file.

Finally FORK_PROCESS commands should be added for any other modules running on the host (such as protocol modules, user's application or diagnostic utilities). For example:

```
FORK_PROCESS      s7_mgt
FORK_PROCESS      upe
```

3.3 Protocol Configuration

The PCCS Development Package contains a protocol configuration utility, **s7_mgt** which will perform initialisation of all the software modules running on the signalling card. It reads the protocol configuration data from a text file called **config.txt** and provides a quick and flexible method of configuring the protocol modules without the need to write any software for that purpose.

Alternatively the protocol stack may be configured by sending the individual configuration messages documented in the per-module Programmer's Manuals to each protocol module. This approach is of particular use when the application needs to reset the board and run a new configuration without stopping the application program. It is described in a later section.

3.3.1 Protocol Configuration using the s7_mgt utility

The default configuration file used by s7_mgt is config.txt. The -k option allows the user to specify an alternative filename if required. For example:

```
s7_mgt -kmyconfig.txt
```

The format of the configuration file commands are described in Appendix A.

The command line option "-d" will cause s7_mgt to display extra diagnostic information which may be useful when making changes to the system configuration.

3.3.2 Protocol Configuration using individual messages

As an alternative to using the s7_mgt configuration utility it is possible to carry out protocol configuration by building and sending messages directly to the board. This approach does mean that it is necessary to write some application code to handle configuration but has the advantage that the application can, if required, re-configure the board without re-starting the application.

All communication with the board is in the form of sending and receiving messages. The configuration sequence is described in the following section. The application should allocate a message structure using the library function **getm()** and send it to the board using the library function **GCT_send()**. The application should periodically call the library function **GCT_receive()** or **GCT_grab()** in order to receive messages from the board. GCT_receive() will block until a message is available whilst GCT_grab() will return immediately. Once the application has finished processing the received message it should release the message structure back to the system by calling the library function **relm().** The library functions are all described in the Software Environment Programmer's Manual.

To configure the board using individual messages the following sequence should be used. (The format of all the messages is described in Section 5 of this manual).

1. Build and send an **SSD Reset Message**. This contains the parameters to initialise the ssd module.

2. Build and send a **Board Reset Message** for each board. This contains the address of the board and the name of the code file. It causes the board to be reset and the code file downloaded.

3. Wait until a **Board Status Message** is received (for each board), inspect the status field to determine whether or not the reset operation was successful. On failure you should check carefully the parameters and try again. On success continue to the next step.

4. Build and send a **Board Configuration Message**. This contains all mandatory protocol configuration parameters (such as point codes, physical link settings and MTP configuration parameters) for the Message Transfer Part (MTP).

5. Wait until a Board Configuration Confirmation Message is received, inspect the status field which will be set to zero on success. On failure re-check configuration parameters and go back to resetting the board.

6. Optionally send **MTP Config Route Messages** for any remote signalling points (other than adjacent signalling points. The route configuration for adjacent signalling points is automatically set up using the board configuration message). Ensure that the status is zero in the confirmation message.

7. If a user part (eg ISUP, TUP, NUP) is included in the code file build and send the per-module configuration message (as described in the Programmer's Manual for the User Part Module). Ensure that the status is zero in the confirmation message.

8. If a user part is included, build and send circuit group configuration messages for each circuit group (as described in the Programmer's Manual for the User Part Module). Ensure that the status is zero in the confirmation message.

9. The protocol stack is now configured ready for use in the same way as if the configuration utility s7_mgt has been used. The user should send an MTP Activate Signalling Link message for each signalling link to start up SS7 operation.

3.4 Using the Septel ISA SCbus

3.4.1 Introduction

The Septel ISA signalling card supports up to two E1 / T1 Line Interface Units and an SCbus interface. The on-board signalling processor handles the SS7 signalling timeslots whilst the remaining circuits (voice or data bearer circuits) are passed to the SCbus for distribution to other cards.

All communication between the application and the board is message based. Initial configuration is usually handled by the configuration utility **s7_mgt**, which takes commands from the text file (config.txt) and generates all the necessary configuration messages for the board. Subsequent operation is entirely message driven, messages being passed in both directions between the board and the application.

One of the roles of the application is to control the dynamic switching between the SCbus and the E1 / T1 line interfaces. This section provides details of how to interface with the SCbus, including the initial (static) configuration and the subsequent (dynamic) switching.

The operation of the SCbus switching interface is described in terms of the Dialogic SCbus switching model using the messages **MVD_SC_DRIVE_LIU**, **MVD_MSG_SC_LISTEN** and **MVD_MSG_SC_PATTERN** and three config.txt commands **LIU_SC_DRIVE**, **SCBUS_LISTEN** and **SCBUS_PATTERN**. These messages and commands are designed specifically to provide a direct mapping to the standard Dialogic SCbus switching model. They are the recommended method for interfacing to the SCbus in all new designs. Previous functionality using the MVD_MSG_SC_CONNECT message continues to be fully supported by the board so existing applications require no modification.

The main use of the DRIVE_LIU and SC_LISTEN commands and messages is to make connections between the line interface units and the SCbus. It is also possible to use these messages to connect signalling channels to the SCbus. Contact DataKinetics directly for more information about this mode of operation.

3.4.2 Switching Model

The basic switching model assumes that at system initialisation all incoming E1 / T1 timeslots and all resource card output timeslots are connected up to channels on the SCbus and that these connections are never changed. This has the advantage that once the on-board SCbus drivers have been set up they are never changed so the chances of inadvertently causing SCbus conflict is minimised. It also means that the user can predict the exact SC channels where any input timeslot can be located and this in turn can assist with fault diagnosis and general system test.

It is also possible to generate fixed patterns on the SCbus and one use of these fixed patterns is to provide the voice idle pattern for presentation to the network on all circuits where there is currently no active calls. This idle pattern should also be set up at system initialisation time.

Having completed the system initialisation all drives to the SCbus are set up. The only operation that needs to happen on a dynamic (call by call) basis is that the E1 / T1 output timeslots and the resource card input timeslots must "Listen" to the appropriate SC channel whenever a new call arrives or a new resource needs to be selected.

When a new call arrives the application will, in general, need to initiate two listen commands. One causing the resource to listen to the appropriate SCbus channel to hear the incoming voice path and the other causing the E1 / T1 interface to listen to the output from the resource card to generate the outgoing voice path.

When a call clears, the operation is virtually identical to that of a call arriving except that both devices are told to listen to the appropriate voice idle pattern on the SCbus.

3.4.3 Static Initialisation

Static initialisation is handled by the s7_mgt utility. For each E1 / T1 line interface unit user should include an **LIU_SC_DRIVE** command in the config.txt file. The syntax for this command is detailed in appendix A.

The **LIU_SC_DRIVE** command has several parameters. **board_id** and **liu_id** together uniquely identify the affected line interface unit. **sc_channel** is the channel number of the first channel on the SCbus that is to be used for timeslots from the specified LIU. **ts_mask** is a timeslot mask identifying which timeslots on the E1 / T1 interface need to be connected to the SCbus. The least significant bit of this mask should always be set to zero - it corresponds to timeslot zero which is not present on T1 streams and is used for framing information on E1 streams.

The user may wish to generate a voice idle pattern on the SCbus for use as the signal output to the network when there is no active call on a circuit. The command **SCBUS_PATTERN** is used for this purpose (see appendix A). It is recommended that an idle pattern be generated for each board so that a failure on one board does not remove the idle pattern used by other boards.

As an example consider a two board system where the first board has 2 E1 ports and the second board has 2 T1 ports. We allow the first 512 SCbus channels to be used by other cards in the system and therefore base our first timeslots at sc_channel 512.

```
LIU_SC_DRIVE 0 0 512 0xfffefffe  * 30 E1 voice ccts on ts 1..15 & 17..31
LIU_SC_DRIVE 0 1 542 0xfffefffe  * 30 E1 voice ccts on ts 1..15 & 17..31
LIU_SC_DRIVE 1 0 572 0x00fffffe  * 23 T1 voice ccts on timeslots 1..23
LIU_SC_DRIVE 1 1 595 0x00fffffe  * 23 T1 voice ccts on timeslots 1..23
```

The next free SCbus channel is 512+30+30+23+23 = 618 so we can use this and the next one for E1 and T1 voice idle patterns respectively:

```
SCBUS_PATTERN  0  0  618  0x2a                * E1 idle pattern
SCBUS_PATTERN  0  0  619  0xff                * T1 idle pattern
```

3.4.4 Dynamic Operation

The application controls dynamic changes to SCbus switching by sending the **MVD_MSG_SC_LISTEN** message to the board. This message is documented in Section 5 of this manual. It contains the **liu_id** (0 or 1), the **timeslot** number on the E1 / T1 interface and the SCbus channel number (**sc_channel**) to which the timeslot should listen. The message is directed to the correct board by calling the **GCT_set_instance** function prior to calling **GCT_send**.

When a new call arrives, the application will need to instigate 2 listen commands (although they will not necessarily both apply to the Septel ISA board). One will connect the voice circuit in the forward direction and the other will connect it in the backward direction.

When a call terminates, the application should again issue 2 listen commands to ensure that the network port and the resource card both see the voice idle pattern.

3.4.5 Example code - Building and sending SC_LISTEN

```
/*
 * Example function for building and sending an MVD_MSG_SC_LISTEN
 * message to a PCCS6 signalling card.
 *
 * The only change that the user needs to make is to fill in the
 * OUR_MOD_ID definition below so that is equal to the module_id
 * of the application module.
 */

#define OUR_MOD_ID   (0xef)

#include "system.h"             /* Definitions of u8, u16 etc */
#include "msg.h"                /* Definitions of HDR, MSG etc */
#include "libc.h"               /* Used only for memset prototype */
#include "sysgct.h"             /* Prototypes for GCT_xxx */
#include "pack.h"               /* Prototypes for rpackbytes */
#include "ss7_inc.h"            /* Message & module definitions */

/*
 * Macro to generate the value for use in the rsp_req field of the
 * message header in order to request a confirmation message:
 */
#define RESPONSE(module)      (((unsigned short) 1) << ((module) & 0x0f))

/*
 * Function to drive an SCbus timeslot
 * onto a timeslot on a PCM port:
 */
int listen_to_scbus(board_id, liu_id, timeslot, sc_channel)
  int board_id;         /* board_id (0, 1, 2 ...) */
  int liu_id;           /* PCM port id (0 or 1) */
  int timeslot;         /* Timeslot on the PCM port (1 .. 31) */
  int sc_channel;       /* SCbus channel number */
{
  MSG   *m;
  u8    *pptr;

  /*
   * Allocate a message (and fill in type, id, rsp_req & len):
   */
  if ((m = getm(MVD_MSG_SC_LISTEN, 0, RESPONSE(OUR_MOD_ID), MVDML_SCLIS)) != 0)
  {
    pptr = get_param(m);
    memset(pptr, 0, m->len);

    /*
     * Enter the parameters in machine independent format:
     */
    rpackbytes(pptr, MVDMO_SCLIS_liu_id, (u32)liu_id, MVDMS_SCLIS_liu_id);
    rpackbytes(pptr, MVDMO_SCLIS_timeslot, (u32)timeslot, MVDMS_SCLIS_timeslot);
    rpackbytes(pptr, MVDMO_SCLIS_sc_channel, (u32)sc_channel, MVDMS_SCLIS_sc_channel);

    m->hdr.dst = MVD_TASK_ID;
    m->hdr.src = OUR_MOD_ID;

    /*
     * Call GCT_set_instance to route the message to the
     * correct board and GCT_send to send the message.
     * If GCT_send returns non-zero release the message.
     */
    GCT_set_instance(board_id, (HDR *)m);
    if (GCT_send(m->hdr.dst, (HDR *)m) != 0)
      relm((HDR *)m);
  }
  return(0);
}
```

4. PROGRAM EXECUTION

This section describes how to start the software running. It assumes that the software has already been installed and the configuration files system.txt and config.txt have been modified accordingly. Refer to previous sections if unsure.

There are three main stages to getting a new application up and running although the precise means of achieving this vary slightly depending upon the operating system.

First the device driver must be installed and run.

Secondly the protocol software running on the host must be run up.

The final stage is to write your application (making use of the examples supplied), compile it (using the header files supplied) and link it with the supplied libraries to generate a finished application program.

The details of how these steps are achieved for each operating system are given below.

4.3 Program execution under Windows NT

Ensure the device driver has been installed using pccsxcfg.exe.

Ensure that the system configuration file (system.txt) has been modified in accordance with system requirements to select the correct protocols etc.

Ensure that the correct code file has been copied into the directory containing all the System7 binaries.

If using s7_mgt, ensure that the protocol configuration file config.txt has been edited to provide correct protocol configuration.

To start the software running, change to directory containing all the binaries and run gctload in the background optionally specifying the system configuration file.

To run the system in a separate console enter:

```
start gctload -csystem.txt &
```

To run the system within the current console enter:

```
gctload -csystem.txt &
```

The **gctload** program will initialise the system environment and start up other processes. The **s7_mgt** process will configure all the protocol modules. A banner will confirm that the system is running.

The example utility mtpsl may be used to activate and deactivate signalling links as follows:

```
mtpsl { act | deact } <linkset_id> <link_ref>

mtpsl act 0 0

mtpsl deact 0 0
```

To shutdown the host software, if the system was run in a separate console the processes can be stopped by pressing CTRL-C. If the system was run in the current console enter CTRL-C to stop gctload and stop the other processes individually using the Task Manager (**TASKMGR.EXE**).

4.3 Program execution under SCO UNIX and Linux

Ensure the device driver has been installed and activated.

Ensure that the system configuration file (system.txt) has been modified in accordance with system requirements to select the correct protocols etc.

Ensure that the correct code file has been copied into the directory containing all the System7 binaries.

If using s7_mgt, ensure that the protocol configuration file config.txt has been edited to provide correct protocol configuration.

To start the software running, change to directory containing all the binaries and run gctload in the background optionally specifying the system configuration file.

```
gctload -cmyfile.txt &
```

The **gctload** program will initialise the system environment and start up other processes. The **s7_mgt** process will configure all the protocol modules. A banner will confirm that the system is running.

The example utility mtpsl may be used to activate and deactivate signalling links as follows:

```
mtpsl { act | deact } <linkset_id> <link_ref>

mtpsl act 0 0

mtpsl deact 0 0
```

To shutdown the host software it is first necessary to determine the process id of the gctload process (using the “ps –e” command) and then use the “kill” command to stop gctload. On shutting down gctload will stop ssd and any other processes it started before terminating itself.

4.4 Program execution under QNX

Ensure the device driver has been installed and activated.

Ensure that the system configuration file (system.txt) has been modified in accordance with system requirements to select the correct protocols etc.

Ensure that the correct code file has been copied into the directory containing all the System7 binaries.

If using s7_mgt, ensure that the protocol configuration file config.txt has been edited to provide correct protocol configuration.

To start the software running, change to directory containing all the binaries and run gctload in the background optionally specifying the system configuration file.

```
gctload -cmyfile.txt &
```

The **gctload** program will initialise the system environment and start up other processes. The **s7_mgt** process will configure all the protocol modules. A banner will confirm that the system is running.

The example utility mtpsl may be used to activate and deactivate signalling links as follows:

```
mtpsl { act | deact } <linkset_id> <link_ref>

mtpsl act 0 0

mtpsl deact 0 0
```

To shutdown the host software use the QNX slay utility to kill gctload. On shutting down gctload will stop ssd and any other processes it started before terminating itself.

```
slay gctload
```

4.5 Developing a user application

The Septel Development Package, along with the User Part Development Package contain all the files to allow the user to develop applications. They contain makefile definitions, C header files (.h) and libraries.

A single definitions file is supplied (for each operating system) which contains the definitions relating to the users own development environment. This file is then included in the make files for all other processes. The user may need to modify this definitions file to ensure that correct paths etc are set up.

The definitions file is one of the following depending on the operating system:

```
makdefs.msc       (SCO UNIX)
makdefs.mqx       (QNX)
makdefs.mnt       (Windows NT)
```

The following library files should be linked with the users application code:

```
gctlib.lib        (Windows NT using Microsoft compiler)
gctlibb.lib       (Windows NT using Borland compiler)
gctlib.lib        (Linux)
unxlib.lib and genlib.lib     (SCO UNIX)
gctlib.lib        (QNX)
```

Some simple example programs are supplied to illustrate the techniques for interfacing to the System7 protocol stack although they are not intended to show a real application. Before starting to develop an application, you should familiarise yourself with the example programs and how they are built.

The example programs are contained on the User Part Development Package.

upe is a framework for a User Part module and contains a worked example of exchanging messages with the MTP3 module. It loops back any MTP-TRANSFER-INDICATIONS messages that it receives and reports MTP indications to the user.

mtpsl is an example of how to send messages to MTP3 to activate and deactivate signalling links. It can be used as a command line tool for this purpose initially. It is intended that eventually the user includes the example code in his own management application.

ctu is an example of how a user application can interface with DataKinetics telephony user parts, eg. ISUP, TUP.

ttu is an example of how a user application can interface with the DataKinetics TCAP protocol module.

A makefile is included to allow you to build the application programs. To build the program change to the src\examples directory and enter (to build ctu):

```
make -f ctu.mak         (SCO UNIX)
make -f ctu.mak         (QNX)
nmake /f ctu.mnt        (Windows NT)
```

5. APPLICATION MESSAGE SPECIFICATIONS

5.1 Overview

This section describes the individual message that may be sent to and received from the board. Some messages are sent by the user's application software whilst others are sent by utility programs such as the System7 configuration utility **s7_mgt**.

Prior to sending any message to the board the application should call the library function GCT_set_instance to select which board the message will be sent to. After receiving a message from the board the application should call the library function GCT_get_instance to determine which board the message came from. These library functions are described in the ***System7 Software Environment Programmer's Manual***.

The messages are grouped into 4 categories: General Configuration Messages, Hardware Control Messages, MTP Interface Messages and Event Indication Messages.

5.1.1 General Configuration Messages

General Configuration Messages are normally issued by the s7_mgt configuration utility in which case they need not and should not be generated by any user application software.

If the user elects not to use s7_mgt then it is necessary for the application to build and send messages to configure the SSD module, reset each individual board, configure each board and optionally configure additional routes.

5.1.2 Hardware Control Messages

Hardware Control Messages are used to control various hardware devices on the board. This includes the E1 / T1 Line Interface Units (LIU), the digital cross connect switches and the clocking mode for the board.

In a static configuration, all these hardware blocks can be set up using the s7_mgt configuration utility along with the appropriate commands in the config.txt file.

If dynamic control of the hardware is required (or the user has elected not to use s7_mgt) then the user application will need to build and send at least some of the Hardware Control Messages.

5.1.3 MTP Interface Messages

MTP Interface Messages allow signalling links to be activated and deactivated by the user and provide a mechanism for communication between the MTP3 module and the user part module (eg. ISUP, TUP, SCCP). In most cases the user part module is part of the System7 software stack so the user does not need to handle the MTP-TRANSFER, MTP-PAUSE, MTP_RESUME & MTP-STATUS primitives as they pass directly between MTP3 and the user part module.

In the case that the user application is implementing the user part functionality then the MTP primitives are applicable and these are documented in the MTP Interface messages section.

5.1.4 Event Indication Messages

Event Indication Messages are the mechanism by which protocol and software error events are reported to the application. These messages are generated asynchronously by different modules within the stack.

5.1.5 Message Type Table

The following table lists, by message type, all the messages described in this manual:

Message Type	Mnemonic	Description
0x0008	MGT_MSG_EVENT_IND	Error Indication
0x0201	MGT_MSG_SS7_STATE	MTP2 Level 2 State Indication
0x0202	MGT_MSG_SS7_EVENT	MTP2 Q.791 Event Indication
0x0301	MGT_MSG_MTP_EVENT	MTP3 Q.791 Event Indication
0x06a0	SSD_MSG_STATE_IND	Board Status Indication
0x0e01	MVD_MSG_LIU_STATUS	LIU Status Indication
0x1e37		Confirmation of LIU_MSG_R_CONFIG
0x1e38		Confirmation of LIU_MSG_R_CONTROL
0x3312		Confirmation of MTP_MSG_CNF_ROUTE
0x3680		Confirmation of SSD_MSG_RESET
0x3681		Confirmation of SSD_MSG_RST_BOARD
0x3e00		Confirmation of MVD_MSG_RESETSWX
0x3e10		Confirmation of MVD_MSG_SETOUTPUT
0x3e16		Confirmation of MVD_MSG_SC_PATTERN
0x3e17		Confirmation of MVD_MSG_SC_LISTEN
0x3e18		Confirmation of MVD_MSG_SC_DRIVE_LIU
0x3e1f		Confirmation of MVD_MSG_SC_CONNECT
0x3e20		Confirmation of MVD_MSG_CNFCLOCK

0x3e30		Confirmation of MVD_MSG_CNFCAR
0x3e34		Confirmation of LIU_MSG_CONFIG
0x3e35		Confirmation of LIU_MSG_CONTROL
0x3f10		Confirmation of MGT_MSG_CONFIG0
0x5e37	LIU_MSG_R_CONFIG	LIU Read Configuration Request
0x5e38	LIU_MSG_R_CONTROL	LIU Read Configuration Request
0x7312	MTP_MSG_CNF_ROUTE	MTP Route Configuration Request
0x7680	SSD_MSG_RESET	SSD Reset Request
0x7681	SSD_MSG_RST_BOARD	Board Reset Request
0x7e00	MVD_MSG_RESETSWX	Reset Switch Request
0x7e10	MVD_MSG_SETOUTPUT	Set Output Request
0x7e16	MVD_MSG_SC_PATTERN	SCbus Pattern Generation Request
0x7e17	MVD_MSG_SC_LISTEN	SCbus Listen Request
0x7e18	MVD_MSG_SC_DRIVE_LIU	SCbus Initialisation Request
0x7e1f	MVD_MSG_SC_CONNECT	SCbus Connect Request
0x7e20	MVD_MSG_CNFCLOCK	Configure Clock Request
0x7e30	MVD_MSG_CNFCAR	Configure Carrier Request (PCCS3 only)
0x7e34	LIU_MSG_CONFIG	LIU Configuration Request
0x7e35	LIU_MSG_CONTROL	LIU Control Request
0x7f10	MGT_MSG_CONFIG0	Board Configuration Request
0x830a		Confirmation of MTP_MSG_ACT_SL
0x830b		Confirmation of MTP_MSG_DEACT_SL
0x8403	MTP_MSG_PAUSE	MTP Pause Indication
0x8404	MTP_MSG_RESUME	MTP Resume Indication
0x8405	MTP_MSG_STATUS	MTP Status Indication
0x8f01	API_MSG_RX_IND	MTP Transfer Indication
0xc30a	MTP_MSG_ACT_SL	MTP Link Activation Request
0xc30b	MTP_MSG_DEACT_SL	MTP Link Deactivation request
0xcf00	API_MSG_TX_REQ	MTP Transfer Request

5.2 General Configuration Messages

5.2.1 SSD Reset Request

Synopsis:

Message sent to SSD once at initialisation to set up run-time options.

NOTE: When using s7_mgt, this message is generated by s7_mgt and should not be generated by the user.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**SSD_MSG_RESET** (0x7680)
id		0
src		Sending module's module_id
dst		**SSD_TASK_ID** (0x20)
rsp_req		used to request a confirmation
hclass		0
status		0
err_info		0
len		24
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	1	module_id - must be set to **SSD_TASK_ID**
1	2	reserved - set to zero
3	1	mgmt_id
4	18	reserved - set to zero
22	2	num_boards

Description:

This message is used during initialisation by the application to reset the ssd module and set up its run-time parameters.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

mgmt_id

The module_id of the management module to which ssd should sent board status indications.

num_boards

The maximum number of boards that ssd will be required to manage. This should not exceed 16.

5.2.2 Board Reset Request

Synopsis:

Message sent to SSD to cause a single board to be reset and a code file downloaded.

NOTE: When using s7_mgt, this message is generated by s7_mgt and should not be generated by the user.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**SSD_MSG_RST_BOARD** (0x7681)
id		board_id
src		Sending module's module_id
dst		**SSD_TASK_ID** (0x20)
rsp_req		used to request a confirmation
hclass		0
status		0
err_info		0
len		24
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	board_type
2	4	phy_id
6	18	code_file

Description:

This message is used during initialisation (or re-configuration) by the application to reset a board and download the code file that contains the operating software for the board.

The download operation is supervised by the device driver which reads the binary format code file and transfers it to the board.

The confirmation message (if requested) will indicate success by a status of 0. This implies that the reset operation has commenced but does not imply completion.

The application should then wait until a **Board Status Indication** is received. This will indicate either successful completion of the reset and download operation or failure during the procedure.

Parameter Description:

board_type

The type of board to be reset. This should be set to 0 for PCCS3 and 1 for PCCS6.

phy_id

The physical id for the board. This field should be set to the same value as the board_id. (ie 0 .. one less than the number of boards supported).

code_file

Null terminated string giving the filename of the code file to be downloaded to the board.

5.2.3 Board Configuration Request

Synopsis:

Message sent to a board immediately after starting the code running to provide protocol configuration parameters.

NOTE: When using s7_mgt, this message is generated by s7_mgt and should not be generated by the user.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MGT_MSG_CONFIG0** (0x7F10)
id	0
src	Sending module's module_id
dst	**MGMT_TASK_ID** (0x8e)
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	56

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	config_type (Must be set to 2)
2	2	flags
4	2	l1_flags
6	2	l2_flags
8	2	max_sif_len
10	2	l3_flags
12	4	pc
16	2	ssf
18	2	up_enable
20	2	link0_flags
22	2	link0_slc
24	4	link0_adj_pc
28	2	link0_stream
30	2	link0_timeslot
32	2	link1_flags
34	2	link1_slc
36	4	link1_adj_pc
40	2	link1_stream
42	2	link1_timeslot
44	2	link2_flags
46	2	link2_slc
48	4	link2_adj_pc
52	2	link2_stream
54	2	link2_timeslot

Description:

This message must be the first message sent to the board once the SS7 software is running. It is used to configure all modules on the board for operation. The message contains signalling point codes for this signalling point and the adjacent signalling point(s), flags to permit various level 1, level 2 and level 3 run-time options to be selected and the physical link parameters.

Once the board has been configured it must not be configured again without first resetting it.

The confirmation message (if requested) will indicate success by a status of 0. To ensure that configuration is complete before subsequent messages are issued to the board, the user should always request a confirmation message and check the status for success.

If the board is not licenced to run the requested software configuration a status value of 0xfe is returned.

Parameter Description:

flags - Global flags

Bit 0 is set to 1 to indicate that the user does not wish to use any signalling software. This allows operation of the board without a software licence button providing the board is only used for E1 / T1 interface and switching purposes. If signalling software is required then this bit must be set to zero.

Bit 1 is set to 1 to cause the E1 / T1 interface to start up in the disabled state (ie. no output signal generated) and remain disabled until the user (or s7_mgt) sends in an LIU Configuration message. When set to zero the E1 / T1 interface port will start up with a default configuration.

Bit 9 is set to 1 to disable automatic MTP route configuration, in which case the user must send individual MTP Route Configuration messages for each destination. When set to zero the board will automatically configure an MTP Route to each adjacent signalling point using the link set directly connected to the signalling point.

Bit 10 is reserved for future use and should always be set to 1.

Bit 12 is set to 1 to cause all signalling links to be automatically activated. Usually this bit is set to zero and the user sends individual MTP Link Activation requests to activate each link.

Bit 15 is set to 1 for diagnostic purposes to cause the results of internal board configuration to be passed to the host. When set, all confirmation messages generated internally on the board during the configuration sequence are sent to the module_id 0xdf on the host.

All other bits are reserved for future use and should be set to zero.

l1_flags - level 1 flags

For **PCCS3** boards only bits 0, 1 and 11 are effective. All other bits should be set to zero. For **Septel ISA (PCCS6)** boards all bits are effective as described below.

Bit 0 is set to 1 to select an external clock reference or zero to use the internal clock source. When set to 1 the setting of bit 2 determines which LIU is used to recover clocks.

Bit 1 is set to 1 to cause a signalling link to be taken out of service if no signalling units are received for an excessive period. If this bit is set to zero then the link will remain in service if continuous flags are received (i.e. no FISUs). Usually this bit should be set to 1.

Bit 2 is only significant when bit 0 is set to 1. It should be set to zero to recover clock from the first PCM port, or 1 to recover clocks from the second PCM port.

Bit 4 is set to 1 for 2MHz SCbus operation, or zero to select 4MHz operation. The use of a 4MHz clock is recommended as it offers twice as many channels on the SCbus.

Bit 6 and **7** together select the SCbus mode according to the following table:

Bit 7	Bit 6	SCbus mode
0	0	The SCbus interface is disabled. This mode should be selected whenever the SCbus module is not fitted or when the user does not wish to use the SCbus.
0	1	**Master** - The board will drive the SCbus clocks.
1	0	**Standby** - The board is configured to take over as SCbus clock master in the event that the SCbus clocks fail.
1	1	**Slave** - The board uses the SCbus clocks (which must be generated by another board on the SCbus).

Bit 9 is used to select monitoring mode for the signalling link. In this mode received MSUs are passed to a host application. For further details of this mode of operation please contact DataKinetics.

Bit 11 is used to select one of two possible 56kbit/s operating modes. It is only effective when the per-link 56kbit/s option is selected and the signalling link is not running on a serial port. When set to 1, 56kbit/s operation does not use bit 8 in the timeslot and when set to zero 56kbit/s operation does not use bit 7 in the timeslot. (The bits of the timeslot are labelled bit 1 .. bit 8 in accordance with ITU-T terminology). Usually this bit should be set to 1.

All other bits are reserved and should be set to zero.

l2_flags - level 2 flags

Bit 1 should be set to 1 for ANSI operation or zero for ITU-T operation.

Bit 3 should be set to 1 for ANSI operation or zero for ITU-T operation.

Bit 5 is set to 1 to cause Link Status Signal Units (LSSU) to have a two octet status field. Otherwise LSSUs will have a single octet status field. Usually this bit should be set to 0.

All other bits are reserved for future use and should be set to zero.

max_sif_len - maximum Signalling Information Field length

The maximum Signalling Information Field length in octets that is permitted over the signalling link. Usually this should be set to 272 although it can also be set to 62 for inter-working to switches that do not support 272 octet messages.

l3_flags - level 3 flags

Bit 0 is set to 1 to disable the level 3 discrimination function (allowing the signalling point to receive all messages irrespective of the destination point code contained in the message) or 0 to allow the discrimination function to function normally.

Bit 1 is set to 1 to disable sub-service field (SSF) discrimination. If this bit is set to 0, received MSUs whose ssf value does not match the configured ssf value will be discarded.

Bit 8 is set to 1 to select ANSI operation or zero for ITU-T operation.

Bit 9 is set to 1 to select ANSI style 24 bit point codes in the MTP routing label or zero to select ITU-T style 14 bit point codes. This bit should always be set to one when ANSI operation is selected.

Bit 10 should be set to 1 for ANSI operation or zero for ITU-T operation.

Bit 11 should be set to 1 for ANSI operation or zero for ITU-T operation.

All other bits are reserved for future use and should be set to zero.

NOTE: For correct ANSI operation bits 8, 9, 10 and 11 must all be set to 1.

pc - point code

The point code of this signalling point coded in pure binary representation. Should be in the range 0 to 16383 for 14 bit point code operation or 0 to 16777215 for 24 bit point code operation.

ssf - sub-service field

The value to be used in the sub-service field of all messages generated by level 3. Should be in the range 0 to 15. For correct ANSI operation the 2 least significant bits must always be set to 1.

up_enable - User Part Enable

A 16 bit mask used to enable or disable reception of messages on a per user part basis. If bit N is set to 1 then messages for user part N will be received by the signalling point.

For example to enable the TUP User Part (Service indicator = 4) set the up_enable field to 0x0010, For ISUP (Service Indicator = 5) set the up_enable field to 0x0020. To use both TUP and ISUP set up_enable to 0x0030.

link*n*_**flags** - Per link flags

Bit 0 is set to 1 to force the use of the emergency proving period during link alignment or 0 to use the appropriate proving period according to Q.703. This bit should usually be set to 0.

Bit 1 is set to 1 to cause a signalling link test (in accordance with ITU-T Q.707) to be carried out before a link is put into service or 0 if a test is not required. This bit should usually be set to 1

Bit 2 is set to 1 to cause a signalling link test (in accordance with ITU-T Q.707) to be carried out every 30 seconds. Note that this bit is ignored unless bit 1 is also set to 1. This bit should usually be set to 1.

Bit 8 is used to select the MTP2 error correction mode. It is set to 1 to select PCR (Preventive Cyclic Retransmission) operation or zero for the Basic Method of Error Correction.

Bit 11 is set to 1 to select 56kbit/s operation for the link or 0 for 64kbit/s operation.

NOTE: When using a serial port, 56kbit/s operation is only supported when the clock is applied externally.

Bit 13 is only used when the link has been configured to run over a serial port (ie. bit 14 is set). If set to 1 an external clock will be used (Receive clock). If set to zero an internal clock (Transmit clock) will be used. If the link has not been configured to run over a serial port, this bit must be set to 0.

Bit 14 is set to 1 to use a serial port rather than a PCM timeslot for this link. In this mode the stream and timeslot parameters for this link will be ignored (and should be set to zero). If this bit is set to zero, the link will use the specified stream and timeslot. The serial port used by the signalling processors for each link is fixed, according to the following table:

link*n*	**Serial Port**
0	Cannot be used for a serial port
1	A
2	B

Bit 15 is set to 1 to disable the link or 0 to enable the link.

All other bits are reserved for future use and should be set to zero.

link*n*_**slc** - Signalling link code.

The signalling link code for the link which must be in the range 0 to 15. The signalling link code must be agreed with the administration at the other end of the link and must be unique within a link set. Usually the first link in a link set is assigned the value 0, the next one and so on.

link*n*_**adj_pc** - Adjacent point code.

The point code of the signalling point at the remote end of the link. Should be in the range 0 to 16383 for 14 bit point code operation or 0 to 16777215 for 24 bit point code operation. (*Note that all links in a link set must have the same adjacent point code*).

link*n*_**stream** - Signalling stream.

This is a reference to the logical 2Mb/s PCM highway from which the signalling processor is to insert the signalling, e.g. 16 to select Line Interface A. For more information on the board data paths, refer to the board User Manual.

link*n*_**timeslot** - Signalling timeslot.

The timeslot used for signalling in the range 0 .. 31. For an E1 interface the valid range is 1 .. 31. For a T1 interface the valid range is 1 .. 24.

5.2.4 MTP Route Configuration Request

Synopsis:

Message sent to MTP3 to add a route to the MTP routing table.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**MTP_MSG_CNF_ROUTE** (0x7312)
id		0
src		Sending module_id
dst		**MTP_TASK_ID**
rsp_req		used to request a confirmation
hclass		0
status		0
err_info		0
len		32
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	4	**dpc** - Destination point code.
4	1	**norm_ls** - linkset_id of normal link set.
5	1	**second_ls** - linkset_id of optional link set.
6	2	Reserved for future use, must be set to zero.
8	2	**flags** - run time options (see below)
10	2	**up_enable** - User part enable mask
12	20	Reserved for future use, must be set to zero.

Description:

This message is used by the application to add routes to the MTP routing table. Each route consists of a destination point code and the linkset_id of one or two link sets over which traffic to the destination should be routed. When two link sets are supplied the user can elect to load share traffic or to treat the link sets as a primary link set and a secondary link set.

When the board is configured (using the Board Configure Request) the routing table is (by default) initialised so that the adjacent signalling point is the only accessible destination for the link set. If this is the only destination point code required then there is no need to use this message.

If it is necessary to provide routing over more than a single link set to a destination then bit 9 in the flags field of the Board Configure Request should be set to one and the user should send in MTP_MSG_CNF_ROUTE messages for each destination (including all adjacent signalling points)

The confirmation message (if requested) will indicate success by a status of 0.

NOTE: A route configuration message must be issued for each adjacent signalling point even if no user part messages are sent to the adjacent signalling point.

Parameter Description:

dpc destination point code

The point code of the remote signalling point for which this message is configuring routing data. It may be either an adjacent point code or a point code accessible via an adjacent Signalling Transfer Point.

norm_ls

The linkset_id of the normal link set used to reach the specified destination. This may be any of the following options:

a) The only link set used to reach the destination.
b) The preferred link set used to reach the destination.
c) One of a pair of links sets forming a combined link set.

In the latter two cases a second link set must also be specified.

Within a link set messages will automatically be load shared across links using the Signalling Link Selection (SLS) field in the message.

second_ls

The linkset_id of an optional second link set used to reach the specified destination. This may be either of the following options:

a) The secondary link set used to reach the destination only on failure of the preferred link set.
b) One of a pair of links sets forming a combined link set over which load sharing will take place. (in this case bit 1 must also be set in the flags field of the message).

When a second link set is specified the user must also set **bit 0** in the flags field of this message.

up_enable

This is a 16 bit field used identify the user parts that are supported over this route. The bits are labelled 0 to 15 and for each user part supported the bit corresponding to the Service Indicator for that user part should be set. (eg. To support just ISUP messages, the ISUP Service Indicator is 5 so bit 5 should be set. Therefore a up_enable value of 0x0020 would be appropriate).

flags

This field is a 16 bit field containing run-time configuration options for the route as follows:

Bit 0 is set to 1 to indicate that a second link set is contained within the message. If zero the second_ls field is ignored.

Bit 1 is used to determine whether or not to load share messages across the two link sets. It is only used when two link sets are specified for the route. When set the MTP3 module will load share messages for the destination equally across each of the two specified link sets. Otherwise the MTP3 module will consider the normal link set to be the preferred link set and will only use the second link set in the event of failure of the normal link set. The bit should be set to 1 to enable load sharing across the two link sets or zero to disable load sharing and use preferred and secondary link sets.

All other bits are reserved for future use and must be set to zero.

5.3 Hardware Control Messages

5.3.1 LIU Configuration Request

Synopsis:

Message sent by the application to establish the operating mode for a Line Interface Unit (LIU).

NOTE: When using s7_mgt, this message is generated by s7_mgt as a result of the LIU_CONFIG command. It therefore need not be generated by the user.

NOTE: This message is NOT supported by the PCCS3

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**LIU_MSG_CONFIG** (0x7e34)
id	liu_id (0 or 1)
src	Sending Module ID
dst	**MVD_TASK_ID** (0x10)
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	40

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	1	liu_type
1	1	line_code
2	1	frame_format
3	1	crc_mode
4	1	build_out
5	1	faw
6	1	nfaw
7	4	Reserved for future use, must be set to zero
11	1	ais_gen
12	1	rai_gen
13	1	idle
14	4	clear_mask
18	22	Reserved for future use, must be set to zero

Description:

This message is sent to the board to configure the operating mode a line interface unit. All configuration parameters must be supplied in the message (it is not possible to modify individual operating parameters in isolation). On receipt of the message the board will first verify that the fitted hardware options support the requested operating mode and will then initialise (or re-initialise) the line interface unit.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

A description of the permitted parameter values are given below. When the board is configured, the line interfaces will be initialised with a default mode based on the hardware type (unless LIU operation is disabled). Where appropriate these default settings are shown below as [E1 default] or [T1 default].

liu_type

The physical type of interface according to the following table: (note that this must be selected by the user to be appropriate for the actual hardware fitted otherwise an error status will be returned).

liu_type	Description
1	Disabled (used to deactivate a LIU). In this mode the LIU will not produce an output signal.
2	E1 75ohm unbalanced interface
3	E1 120ohm balanced interface
4	T1
5	E1 75ohm or 120ohm setting based on fitted hardware

line_code

The line coding technique taken from the following table:

line_code	Description
1	HDB3 (E1 only) [E1 default]
2	AMI with no Zero Code Suppression
3	AMI with Zero Code Suppression (The appropriate bit in the clear_mask parameter may be set to disable Zero Code Suppression for individual timeslots if required) (T1 only)
4	B8ZS (T1 only) [T1 default]

frame_format

The frame format taken from the following table:

frame_format	Description
1	E1 double frame (E1 only) [E1 default]
2	E1 CRC4 multiframe (E1 only)
3	F4 (T1 only)
4	D3/D4 (Yellow alarm = bit 2 in each channel) (T1 only) [T1 default]
5	D3/D4 (Yellow alarm = last bit of multiframe) (T1 only)
6	ESF (Yellow alarm = bit 2 in each channel) (T1 only)
7	ESF (Yellow alarm in data link channel) (T1 only)
8	SLC 96 (T1 only)

crc_mode

The CRC mode taken from the following table:

crc_mode	Description
1	CRC generation disabled [E1 default][T1 default]
2	CRC4 enabled (E1 only)
3	CRC4 compatibility mode (E1 only)
4	CRC6 enabled (T1 only)

build_out

Configurable line build out is not supported by the PCCS6 so the following fixed values must be used.

build_out	Description
0	Setting for E1 devices [E1 default]
1	Setting for T1 devices [T1 default]

faw

The 8 bit value to be used for any E1 frame alignment word bit positions that are not modified by other options. This allows the spare bit designated ‘For International Use’ to be set by the user when CRC4 mode is disabled. Valid values are 0x9b or 0x1b. When using T1 this parameter should be set to zero. [E1 default = 0x9b].

nfaw

The 8 bit value to be used for any E1 non-frame alignment word bit positions that are not modified by other options. Normally this parameter should be set to 0x9f. When using T1 this parameter should be set to zero. [E1 default = 0x9f].

ais_gen

The (initial) mode used to generate the Alarm Indication Signal (Blue Alarm) taken from the following table. The user may subsequently modify the setting of the outgoing signal using the LIU_MSG_CONTROL message.

ais_gen	Description
1	Disabled - do not generate AIS / Blue alarm [E1 & T1 default]
2	Enabled - generate AIS / Blue alarm

rai_gen

The (initial) mode used to generate the Remote Alarm Indication (Yellow Alarm) taken from the following table. The user may subsequently modify the setting of the outgoing RAI alarm using the LIU_MSG_CONTROL message.

rai_gen	Description
1	Disabled - do not generate RAI / Yellow alarm
2	Forced active - generate RAI / Yellow alarm
3	Automatic generation of RAI / Yellow alarm upon loss of synchronisation [E1 default] [T1 default]
4	Automatic generation of RAI / Yellow alarm upon loss of synchronisation or BER > 1 in 1000 (E1 only)

idle

The 8 bit idle pattern for use on this LIU. Currently this only takes affect when a channel loop back is applied using the LIU_MSG_CONTROL message.

clear_mask

For use with T1 interfaces and line_code mode 3 (AMI with Zero Code Suppression) to disable zero code suppression on selected channels. This parameter is a 32 bit mask. Zero code suppression should always be disabled for the signalling channel timeslot by setting the appropriate bit in the mask. The least significant bit corresponds to timeslot 0 and the most significant bit to timeslot 31. Bits should be set to 1 to disable zero code suppression.

5.3.2 LIU Control Request

Synopsis:

Message sent by the application to dynamically control operation for a Line Interface Unit (LIU). Allows setting of outgoing alarms and diagnostic loopbacks.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**LIU_MSG_CONTROL** (0x7e35)
id	liu_id (0 or 1)
src	Sending Module ID
dst	**MVD_TASK_ID** (0x10)
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	16

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	1	ais_gen
1	1	rai_gen
2	1	loop_mode
3	1	loop_channel
4	12	Reserved for future use, must be set to zero

Description:

This message is sent to the board to perform dynamic changes to the operation of the Line Interface Unit. It allows the user to control generation of AIS (Blue alarm) and RAI (Yellow alarm) and to activate various diagnostic loopback modes.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

ais_gen

The mode used to generate the Alarm Indication Signal (Blue Alarm) taken from the following table:

ais_gen	Description
0	Do not change AIS / Blue alarm generation mode
1	Disabled - do not generate AIS / Blue alarm
2	Enabled - generate AIS / Blue alarm

rai_gen

The mode used to generate the Remote Alarm Indication (Yellow Alarm) taken from the following table:

rai_gen	Description
0	Do not change RAI / Yellow alarm generation mode
1	Disabled - do not generate RAI / Yellow alarm
2	Forced active - generate RAI / Yellow alarm
3	Automatic generation of RAI / Yellow alarm upon loss of synchronisation
4	Automatic generation of RAI / Yellow alarm upon loss of synchronisation or BER > 1 in 1000. (Defaults to rai_gen mode 3 if hardware does not support BER > 1 in 1000 detection)(E1 only)

loop_mode

The diagnostic loop back mode taken from the following table:

loop_mode	Description
0	Do not change diagnostic loop back mode
1	Disabled - remove any diagnostic loop
2	Payload loopback
3	Remote loopback
4	Local loopback
5	Channel loopback (on channel specified in loop_channel parameter)

loop_channel

The channel number for the channel loopback operation.

5.3.3 LIU Read Configuration Request

Synopsis:

Message sent by the application to read back the current LIU configuration from the board.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**LIU_MSG_R_CONFIG** (0x5e37)
id	liu_id (0 or 1)
src	Sending Module ID
dst	**MVD_TASK_ID** (0x10)
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	40

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	40	Parameter area formatted as for as the LIU_MSG_CONFIG message. The user should set the fields to zero and the module will write the current configuration parameters in the confirmation message.

Description:

This message is sent to the board to read back the current operating configuration of the Line Interface Unit.

The user should always request a confirmation message. This will indicate success by a status of 0 and will contain the current configuration parameters in the parameter area of the message.

5.3.4 LIU Read Control Request

Synopsis:

Message sent by the application to read back the current LIU control options from the board.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**LIU_MSG_R_CONTROL** (0x5e38)
id	liu_id (0 or 1)
src	Sending Module ID
dst	**MVD_TASK_ID** (0x10)
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	16

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	16	Parameter area formatted as for as the LIU_MSG_CONTROL message. The user should set the fields to zero and the module will write the current control parameters in the confirmation message.

Description:

This message is sent to the board to read back the current control parameters selected for the Line Interface Unit.

The user should always request a confirmation message. This will indicate success by a status of 0 and will contain the current control parameters in the parameter area of the message.

5.3.5 LIU SCbus Initialisation Request

Synopsis:

This message is sent to the board at initialisation time to set up a static switch path through the card between the Line Interface Unit (LIU) and the SCbus. It connects selected incoming voice timeslots from an E1/T1 LIU to a sequential block of channels on the SCbus and prepares the outgoing timeslots for subsequent use by the MVD_MSG_SC_LISTEN message.

NOTE: When using s7_mgt, this message is generated by s7_mgt as a result of the LIU_SC_DRIVE command. It therefore need not be generated by the user.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**MVD_MSG_SC_DRIVE_LIU** (0x7e18)
id		0
src		Sending Module ID
dst		**MVD_TASK_ID** (0x10)
rsp_req		used to request a confirmation
hclass		0
status		0
err_info		0
len		10
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	**liu_id**
2	2	**sc_channel**
4	4	**ts_mask**
8	2	**mode**

Parameter Description:

liu_id

The identifier of the E1/T1 Line Interface Unit. This should be set to 0 for the first LIU and 1 for the second LIU.

It is also possible to use this command to connect the signalling processor timeslots to the SCbus. Contact DataKinetics for more information about this mode of operation.

sc_channel

The channel number of the first channel to be used on the SCbus. This should be in the range from 0 up to one less than the total number of channels on the SCbus. In the case of a 4MHz SCbus the valid range for sc_channel is 0 .. 1023.

When deciding what value to use for sc_channel the user should note that, for example, Dialogic cards usually allocate channels on the SCbus commencing at the lowest numbered channels and working up. It is therefore recommended that the user should determine the maximum number of sc_channels that will be used by any Dialogic boards in the system (allowing for future expansion) and set the sc_channel to the next value.

ts_mask

A 32 bit timeslot mask where each bit position is set to 1 if the corresponding timeslot on the E1/T1 interface is required to be connected to the SCbus. The least significant bit (bit 0) represents timeslot 0. Each timeslot for which the corresponding bit is set in **ts_mask** will be connected up to the SCbus, other timeslots will not be affected in any way.

Timeslots containing SS7 signalling that will be processed by the signalling processor on the card should not be included in the timeslot mask. Usually the mask should be set to include all bearer (voice) timeslots but no signalling timeslots. Bit 0 (corresponding to timeslot 0 on the LIU) will not usually be set as timeslot 0 for an E1 interface contains synchronisation information whilst timeslot 0 for a T1 interface does not even exist.

As an example, for an E1 interface with SS7 signalling on timeslot 16, and the remaining 30 timeslots used for voice circuits, ts_mask should be set to the value 0xfffefffe. For a T1 interface with signalling on timeslot 24, ts_mask should be set to the value 0x00fffffe

mode

This parameter controls how the SCbus channels are allocated. Usually (**mode=1**) the first timeslot connected to the SCbus is connected to **sc_channel** and each subsequent timeslot that is connected will be connected to the next SCbus channel. This allows maximum utilisation of channels on the SCbus and is compatible with channel allocation by Dialogic cards.

An alternative mode (**mode=2**) (that should only be used if there is a specific requirement for it) associates (but does not necessarily connect) timeslot 0 on the LIU with **sc_channel** and subsequent timeslots on the LIU with subsequent SCbus channels. Connections are only made when the corresponding bit in the timeslot mask is set to 1. This mode of operation preserves the spacing between timeslots that was originally found on the E1/T1 interface but does result in a number of SCbus channels being not used.

5.3.6 SCbus Listen Request

Synopsis:

Message sent to the board to establish a connection from the SCbus to an outgoing timeslot on the PCCS6 card Line Interface Unit (LIU).

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MVD_MSG_SC_LISTEN** (0x7e17)
id	0
src	Sending Module ID
dst	**MVD_TASK_ID** (0x10)
rsp_req	Used to request a confirmation
hclass	0
status	0
err_info	0
len	6

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	liu_id
2	2	timeslot
4	2	sc_channel

Description:

This message is sent to the board to establish a connection from the SCbus to an outgoing timeslot on the E1 / T1 Line Interface Unit (LIU). It is issued by the application and is typically used at the start and end of each call although it may also be issued during a call to connect to a different resource.

Correct operation of this message is dependent upon the use, at initialisation time, of the MVD_MSG_SC_DRIVE_LIU message (or the LIU_SC_DRIVE command in config.txt when using s7_mgt).

When a new call arrives the application should use this message to connect the appropriate resource from the SCbus out to the network. When the call finishes the application should again use this message but this time to 'listen' to the appropriate IDLE pattern on a different SCbus channel. (Refer to the message MVD_MSG_SC_PATTERN for details of generating a fixed pattern on the SCbus).

The MVD_MSG_SC_LISTEN message can also be generated at configuration time using s7_mgt as a result of the SCBUS_LISTEN command in the config.txt file. However, this will of course only set up a static configuration and will still require the user application to control any dynamic connections.

Parameter Description:

liu_id

The identifier of the E1/T1 Line Interface Unit. This should be set to 0 for the first LIU and 1 for the second LIU.

timeslot

The timeslot number on the E1/T1 line interface unit on which the data from the SCbus will be transmitted. The valid range for **timeslot** is 1 to 31 for an E1 interface and 1 to 24 for a T1 interface. The timeslot must have been enabled by setting the appropriate bit in the **ts_mask** when the MVD_MSG_SC_DRIVE_LIU message was originally issued.

sc_channel

The channel number on the SCbus to which the LIU will listen. This should be in the range from 0 up to one less than the total number of channels on the SCbus. In the case of a 4MHz SCbus the valid range for sc_channel is 0 .. 1023.

5.3.7 SCbus Pattern Request

Synopsis:

Message sent to the board in order to generate a fixed pattern on a specific SCbus channel.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MVD_MSG_SC_PATTERN** (0x7e16)
id	0
src	Sending Module ID
dst	**MVD_TASK_ID** (0x10)
rsp_req	Used to request a confirmation
hclass	0
status	0
err_info	0
len	6

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	pattern_id
2	2	sc_channel
4	2	pattern

Description:

This message is sent to the board in order to generate a fixed pattern on a specific SCbus channel. It is typically issued at initialisation time to generate an IDLE pattern that can subsequently be 'listened' to by any LIU timeslots that are not carrying active calls.

The Septel ISA board allows for generation of up to 32 different patterns and each pattern may be set to any 8 bit value by the user. Usually only a few patterns are required per-board.

The MVD_MSG_SC_PATTERN message can also be generated at configuration time using s7_mgt as a result of the SCBUS_PATTERN command in the config.txt file.

Parameter Description:

pattern_id

The logical identifier for the pattern generator used to generate the pattern. This must be in the range 0 to 31 and would usually be set to 0 for the first pattern, 1 for the next and so on.

sc_channel

The channel number on the SCbus where the pattern will be generated. This should be in the range from 0 up to one less than the total number of channels on the SCbus. In the case of a 4MHz SCbus the valid range for sc_channel is 0 .. 1023.

pattern

The value of the pattern data to be generated on the SCbus. The value should be in the range 0 to 255. Typically the value might be 0xff for an all ones idle pattern or 0x2a for an ITU-T E1 idle pattern.

5.3.8 Reset Switch Request

Synopsis:

Resets the digital switch to its default state in accordance with the current board configuration.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MVD_MSG_RESETSWX** (0x7e00)
id	0
src	Sending Module ID
dst	**MVD_TASK_ID** (0x10)
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	0

Description:

This message is sent to the board to reset the state of the digital cross connect switch in accordance with the configuration set using the board configuration message. All MVIP and SCbus streams are tristated and Network connections are set to the proper conditioned idle state.

When an SCbus module is fitted, this message also serves to reset the SCbus interface, removing any connections made on the SCbus.

The confirmation message (if requested) will indicate success by a status of 0. On receipt of the confirmation message the operation to reset the switch will have completed.

5.3.9 Set Output Request

Synopsis:

Message sent to the board to control the switch path through the MVIP switch.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**MVD_MSG_SETOUTPUT** (0x7e10)
id		0
src		Sending Module ID
dst		**MVD_TASK_ID**
rsp_req		used to request a confirmation
hclass		0
status		0
err_info		0
len		12
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	output_stream
2	2	output_slot
4	2	mode
6	2	input_stream (only used if mode = 2)
8	2	input_slot (only used if mode = 2)
10	2	message (only used if mode = 1)

Description:

This message is sent to the board to control the state of an output timeslot on the digital switch. The output can be either disabled (ie. made tristate), assigned a fixed 8 bit pattern or sourced from one of the switch input streams.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

output_stream

Reference to the 2 Mbit/s stream for the output of the cross-connection or the fixed data. Logical mappings for the PCCS3 and Septel ISA (PCCS6) are shown in the tables below.

output_slot

Timeslot in the range 0 -31 within the output stream.

mode

0 = disable the output timeslot
1 = output a fixed pattern on the output timeslot
2 = connect the input timeslot to the output timeslot
3 = make a duplex connection between the input timeslot and output timeslot

input_stream

Reference to the 2 Mbit/s stream for the input of the cross-connection (only used when mode = 2, otherwise it should be set to zero). Logical mappings for the PCCS3 and Septel ISA (PCCS6) are shown in the tables below.

input_slot

Timeslot in the range 0 - 31 within the input stream (only used when mode = 2 or mode = 3, otherwise should be set to zero).

message

Fixed data pattern to be output when **mode** = 1. If **mode** is not set to 1 then **message** should be set to zero.

Note: When using a PCCS3 board in T1 mode, a mapping is required between the timeslot specified and the timeslot on the T1 trunk, as shown in the following table. For T1 operation with a PCCS6 or E1 operation with either board, no mapping is required.

T1 Channel	input-slot/ output-slot
1	1
2	2
3	3
4	5
5	6
6	7
7	9
8	10
9	11
10	13
11	14
12	15

T1 Channel	input-slot/ output-slot
13	17
14	18
15	19
16	21
17	22
18	23
19	25
20	26
21	27
22	29
23	30
24	31

The following table details the stream assignments for the **PCCS3** input_stream and output_stream:

Stream ID	Output stream	Input stream	Notes
0	O0	I0	Corresponds to cross-connect switch output O0 and input I0 as described in the PCCS3 User Manual. By default the MVIP wire-wrap matrix connects O0 to MVIP stream DSi0 and I0 to MVIP stream DSo0
1	O1	I1	Corresponds to cross-connect switch output O1 and input I1 as described in the PCCS3 User Manual. By default the MVIP wire-wrap matrix connects O1 to MVIP stream DSi1 and I1 to MVIP stream DSo1
2	O2	I2	Corresponds to cross-connect switch output O2 and input I2 as described in the PCCS3 User Manual. By default the MVIP wire-wrap matrix connects O2 to MVIP stream DSi2 and I2 to MVIP stream DSo2
3	O3	I3	Corresponds to cross-connect switch output O3 and input I3 as described in the PCCS3 User Manual. By default the MVIP wire-wrap matrix connects O3 to MVIP stream DSi3 and I3 to MVIP stream DSo3
16	Line Interface	Line Interface	

The following table details the stream assignments for the **PCCS6** input_stream and output_stream:

Stream ID	Output stream	Input stream	
0	MVIP DSi0	MVIP DSo0	
1	MVIP DSi1	MVIP DSo1	
2	MVIP DSi2	MVIP DSo2	
3	MVIP DSi3	MVIP DSo3	
4	MVIP DSi4	MVIP DSo4	
5	MVIP DSi5	MVIP DSo5	
6	MVIP DSi6	MVIP DSo6	Not valid when SCbus module fitted
7	MVIP DSi7	MVIP DSo7	Not valid when SCbus module fitted
8	MVIP DSo0	MVIP DSi0	
9	MVIP DSo1	MVIP DSi1	
10	MVIP DSo2	MVIP DSi2	
11	MVIP DSo3	MVIP DSi3	
12	MVIP DSo4	MVIP DSi4	
13	MVIP DSo5	MVIP DSi5	
14	MVIP DSo6	MVIP DSi6	Not valid when SCbus module fitted
	Inter-switch stream A	Inter-switch stream A	Valid only when SCbus module fitted
15	MVIP DSo7	MVIP DSi7	Not valid when SCbus module fitted
	Inter-switch stream B	Inter-switch stream B	Valid only when SCbus module fitted
16	Line Interface 1	Line Interface 1	
17	Line Interface 2	Line Interface 2	
18	Inter-switch stream C	Inter-switch stream C	Valid only when SCbus module fitted

5.3.10 SCbus Connect Request

Synopsis:

Message sent to the board to control the switch path through the SCbus switch.

Note: This message is supported for backward compatibility only and is not recommended for use in new designs.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**MVD_MSG_SC_CONNECT** (0x7e1f)
id		0
src		Sending Module ID
dst		**MVD_TASK_ID** (0x10)
rsp_req		used to request a confirmation
hclass		0
status		0
err_info		0
len		16
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	local_stream
2	2	local_slot
4	2	mode
6	2	source_stream
8	2	source_slot
10	2	dest_stream
12	2	dest_slot
14	2	pattern

Description:

This message is sent to the board to control the SCbus switch. Seven different actions can be performed depending on the value of the *mode* parameter, these are SCbus to local bus connection and disconnection, local bus to SCbus connection and disconnection, duplex connection and disconnection between SCbus and local bus and source a fixed pattern onto the SCbus. The duplex connection request takes a duplex local bus stream and timeslot and connects it to 2 simplex SCbus stream timeslots.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

The following table depicts which parameters are required for each of the seven different modes. (* = parameter is required)

Mode	Required Parameters						
	local st	local ts	source st	source ts	dest st	dest ts	pattern
1	*	*	*	*			
2	*	*			*	*	
3	*	*	*	*	*	*	
4	*	*					
5	*	*					
6	*	*					
7	*	*			*	*	*

If a parameter is not required it should be set to zero.

local_stream

The *local stream* defines which SCbus local stream to use for all the modes of operation. The SCbus local streams are connected to the inter-switch streams and SCbus pattern generator as follows:

SCbus Local Stream	Connected to
0	Inter-switch stream A
1	Inter-switch stream B
2	Inter-switch stream C
3	Pattern Generator

Refer to the Septel ISA (PCCS6) User Manual for more information on the board data paths.

local_slot

The *local slot* defines which timeslot on the local stream to use for all the modes of operation. The local slot value has the following valid ranges depending on the type of local stream :

Local Stream type	Local Slot range
Local stream to E1 LIU	1…31
Local stream to T1 LIU	1…24
Local stream to MVIP switch block	0…31
Local stream for pattern generation	0…31

mode

The value of the *mode* parameter determines which of the following seven operations to perform.

***mode* = 1** : Make a simplex connection from a timeslot on the SCbus to a timeslot on the local bus. Using parameters local_stream, local_slot, source_stream and source_slot to specify the local and SCbus timeslots respectively.

***mode* = 2** : Make a simplex connection from a timeslot on the local bus to a timeslot on the SCbus. Using parameters local_stream, local_slot, dest_stream and dest_slot to specify the local and SCbus timeslots respectively.

***mode* = 3** : Make a duplex connection between a local stream timeslot and 2 SC bus timeslots. Using parameters local_stream, local_slot, source_stream and source_slot to specify one simplex connection and local_stream, local_slot, dest_stream and dest_slot to specify the other simplex connection.

***mode* = 4** : Remove a simplex connection from a timeslot on the SCbus to a timeslot on the local bus. Using parameters local_stream and local_slot to specify the timeslot for disconnection.

***mode* = 5** : Remove a simplex connection from a timeslot on the local bus to a timeslot on the SCbus. Using parameters local_stream and local_slot to specify the timeslot for disconnection.

***mode* = 6** : Remove a duplex connection between 2 timeslots on the SCbus and 1 timeslot on the local bus. Using parameters local_stream and local_slot to specify both timeslots for disconnection.

***mode* = 7** : Drive the fixed 8 bit pattern defined in *pattern* onto the SCbus timeslot referenced by *destination stream* and *destination slot*. The pattern is stored in the switch routing memory referenced by *local stream* and *local slot*. Owing to the design of the switch it is only possible to source a pattern onto the SCbus. If a pattern is required on a local stream 2 commands must be sent, the first to source a pattern onto the SCbus and the second to connect this pattern to the local stream.

source_stream

The *source stream* references which of the 16 SCbus streams should be used as a source for the data. The parameter takes values in the range 0…15.

source_slot

The *source slot* references the SCbus timeslot from which to connect or disconnect to the local stream. The source slot value has the following ranges depending on the SCbus speed.

SCbus speed	Source Slot range
2 Mb/s SCbus stream	0 .. 31
4 Mb/s SCbus stream	0 .. 63

dest_stream

The *destination stream* references which of the 16 SCbus streams should be used as a destination for the data. The parameter takes values in the range 0…15.

dest_slot

The *destination slot* references the SCbus timeslot to which a local stream timeslot can be connected or disconnected. The destination slot value has the same range as the *source slot*.

pattern

When mode = 7 the 8 bit value *pattern* is driven onto the SCbus timeslot referenced by *dest stream* and *dest slot*. The value pattern is stored in the time space switch memory location referenced by local stream and local slot.

5.3.11 Configure Clock Request (PCCS3)

Synopsis:

Message sent to a PCCS3 board to configure the clock reference source and the clock output options.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**MVD_MSG_CNFCLOCK** (0x7e20)
id		0
src		Sending Module ID
dst		**MVD_TASK_ID**
rsp_req		used to request a confirmation
hclass		0
status		0
err_info		0
len		2
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	clock_mode (See below)

Description:

This message is used to control the PCCS3 on-board clock circuit. It allows the user to select the reference clock source. It also allows the user to select whether or not the MVIP clocks and the MVIP secondary clock signal are driven or not.

Note: The clock mode will be initialised to either 3 or 4 when the board is configured, depending on the selection of either internal clock or external clock respectively in the board configuration message. In many applications this will be sufficient and there will be no need to use the ***Configure Clock Request*** *message.*

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

clock_mode

The following table shows the permissible values:

clock_mode (hex)	reference source	trunk interface	MVIP /F0 /C4 C2	MVIP SEC8K
0001	MVIP /F0	driven	tristate	tristate
0002	MVIP SEC8K	driven	tristate	tristate
0003	PCCS3 local	driven	tristate	tristate
0004	Trunk interface	driven	tristate	tristate
0102	MVIP SEC8K	driven	driven	tristate
0103	PCCS3 local	driven	driven	tristate
0104	Trunk interface	driven	driven	tristate
0201	MVIP /F0	driven	tristate	driven
0203	PCCS3 local	driven	tristate	driven
0204	Trunk interface	driven	tristate	driven

5.3.12 Configure Clock Request (Septel ISA PCCS6)

Synopsis:

Message sent to a Septel ISA (PCCS6) board to configure the clocking mode for the board.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MVD_MSG_CNFCLOCK** (0x7e20)
id	0
src	Sending Module ID
dst	**MVD_TASK_ID**
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	8

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	sc_speed
2	2	clk_mode
4	2	pll_src
6	2	s8k_mode

Description:

This message is used to control the on-board clock circuitry. It allows the user to select the SCbus speed, SCbus mode, MVIP bus mode and the reference clock sources for the PLL and secondary 8k. It allows the user to select between the three modes of SCbus clock driver: master, standby and slave, and to optionally drive or recover MVIP clocks.

The Septel ISA (PCCS6) User Manual provides an overview of the board clock paths.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

sc_speed

This parameter is used to set the SCbus speed, the permissible values are as follows:

Value	SCbus speed
0	No change
1	2.048 MHz
2	4.096 MHz

clk_mode

This parameter determines the clocking mode for the board, the permissible values are as follows:

Value	Clock Mode
0	No change
1	SCbus master
2	SCbus standby
3	SCbus slave
4	Disable SCbus, use PLL clock
5	SCbus master + drive MVIP clocks
6	SCbus standby + drive MVIP clocks
7	SCbus slave + drive MVIP clocks
8	Disable SCbus, use PLL clock +drive MVIP clocks
9	Disable SCbus, use MVIP clocks

When **Disable SCbus** is selected the SCbus interface will be completely disabled. The on board clocks can then be driven either by the phase locked loop (PLL) (in which case the MVIP bus clocks can optionally be driven as well) or recovered from the MVIP bus. One of these modes must be used whenever no SCbus module is fitted.

When **SCbus master** is selected the board will drive SCbus clocks. When **SCbus standby** is selected the board will not drive SCbus clocks unless there is an SCbus clock failure in which case it will automatically take over and drive SCbus clocks. When **SCbus slave** is selected the board will not drive SCbus clocks. In all three cases the MVIP clocks can optionally be driven by the board. The MVIP clocks will be phase locked to the SCbus clocks.

pll_src

This parameter determines the source of the PLL reference clock, the permissible values are as follows:

Value	PLL clock source
0	No change
1	Recovered clock from LIU A
2	Recovered clock from LIU B
5	Local reference oscillator
6	MVIP secondary 8khz signal

s8k_mode

This parameter determines the operating mode of the MVIP secondary 8 kHz clock. The permissible values are as follows:

Value	Secondary 8kHz clock mode
0	No Change
1	SEC8K sourced from LIU A
2	SEC8K sourced from LIU B
6	SEC8K Tristate (ie Disabled)

5.3.13 Configure Carrier Request

Synopsis:

Configures the digital trunk interface and allows alarms to be set and cleared.

NOTE: This message is supported for the PCCS3 board only.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MVD_MSG_CNFCAR** (0x7e30)
id	**liu_id**
src	Sending Module ID
dst	**MVD_TASK_ID**
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	12

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	2	Must be set to 12
2	2	Must be set to 0
4	2	0= CEPT; 1 = T1
6	2	cnf1
8	2	cnf2
10	2	cnf3

Description:

This message is sent to the board to configure the digital trunk interface.

NOTE: If the default settings are acceptable then there is no need to issue this message as the trunk will already be conditioned to this state.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

For E1 (CEPT) operation refer to the table below:

bit no	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnf1	1	0	1	1	Note 1	0	0	0	1	Note 2	1	1	1	1	1	1
cnf2	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
cnf3	Note 3	1	Note 4	Note 5	Note 5	Note 5	Note 5	Note 5	Note 6	0	0	1	1	0	1	1

Note 1 0 = normal operation **(default)**; 1= transmit AIS.

Note 2 0 = normal operation **(default)**; 1= loop back timeslot 16.

Note 3 international bit - value to transmit in bit position 1 of timeslot 0 in non-frame alignment frames **(default = 1)**

Note 4 remote alarm - value to transmit in bit position 3 of timeslot 0 in non-frame alignment frames **(default = 0)**

Note 5 Bits 12 to 8 are the national bits which will be transmitted in bits 4 to 8 respectively of non-frame alignment frames **(default = 1).**

Note 6 international bit - value to transmit in bit position 1 of timeslot 0 in frame alignment frames **(default = 1)**

All other bits **must** be set to the values indicated in the table to ensure correct operation of the board.

For T1 operation refer to the table below:

bit no	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnf1	0	0	Note 1	Note 2	0	0	0	0	1	Note 3	Note 4	0	0	0	1	0
cnf2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cnf3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note 1 0 = normal operation **(default)**
1 = transmit all 1's

Note 2 0 = Super Frame **(default)**
1 = Extended Super Frame

Note 3 0 = Enable zero code suppression **(default)**
1 = Disable zero Code Suppression.

Note 4 0 = Jammed Bit zero suppression
1 = B8ZS zero suppression **(default)**

All other bits **must** be set to the values indicated in the table to ensure correct operation of the board.

5.4 MTP Interface Messages

5.4.1 MTP Link Activation Request

Synopsis:

Primitive request sent to MTP3 module to activate an SS7 signalling link.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MTP_MSG_ACT_SL** (0xc30a)
id	**linkset_id** * 256 + **link_ref**
src	Sending Module ID
dst	**MTP_TASK_ID**
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	0

Description:

This message is issued to the level 3 module to cause a signalling link to be activated in accordance with the procedures in Q.704.

The confirmation message (if requested) will indicate success by a status of 0. This implies that the request for link activation has been received by MTP3 and that activation has commenced.

Note: Receipt of the confirmation message does NOT imply that the link is available for signalling, only that the activation procedure has commenced.

Parameter Description:

linkset_id The link set identifier

Identifies the link set in which the link is to be activated.

link_ref The link reference

Identifies which link in the link set is to be activated.

5.4.2 MTP Link Deactivation Request

Synopsis:

Primitive request sent to MTP3 3 module to deactivate an SS7 signalling link.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MTP_MSG_DEACT_SL** (0xc30b)
id	**linkset_id** * 256 + **link_ref**
src	Sending Module ID
dst	**MTP_TASK_ID**
rsp_req	used to request a confirmation
hclass	0
status	0
err_info	0
len	0

Description:

This message is issued to the level 3 module to cause a signalling link to be deactivated in accordance with the procedures in Q.704.

The confirmation message (if requested) will indicate success by a status of 0.

Parameter Description:

linkset_id The link set identifier

Identifies the link set containing the link to be deactivated.

link_ref The link reference

Identifies which link is to be deactivated.

5.4.3 MTP Transfer Request

Synopsis:

Primitive request sent to MTP3 to convey a Message Signal Unit (MSU) to the board for transmission.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**API_MSG_TX_REQ** (0xcf00)
id	user_part_reference
src	Sending module ID
dst	**MTP_TASK_ID**
rsp_req	0
hclass	0
status	0
err_info	0
len	Number of MSU octets in parameter area

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	len	MSU data in binary format ready for transmission, commencing with the Service Information Octet (SIO) and followed directly by the Signalling Information Field (SIF). The level 2 information (ie BIB, BSN, FIB, FSN LI and FCS) is added by the board.

Description:

This message is used to convey Message Signal Units from the user part to the Message Transfer Part (MTP) on the board for subsequent transmission into the SS7 Network.

The user part should only issue messages to destination point codes for which an MTP Resume indication has been received, otherwise no route is currently available to the destination and the message will be discarded by the Message Transfer Part in accordance with Q.704.

No confirmation messages are issued.

5.4.4 MTP Transfer Indication

Synopsis:

Primitive indication issued by MTP3 to convey a Message Signal Unit (MSU) from the board to the appropriate user part.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**API_MSG_RX_IND** (0x8f01)
id		user_part_reference
src		**MTP_TASK_ID**
dst		User Part Module ID
rsp_req		0
hclass		0
status		0
err_info		0
len		Number of MSU octets in parameter area
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	len	MSU data in binary format as received from the network, commencing with the Service Information Octet (SIO) and followed directly by the Signalling Information Field (SIF). The level 2 information (ie BIB, BSN, FIB, FSN LI and FCS) is removed by the board.

Description:

This message is used to convey Message Signal Units received from the network from the user part to the Message Transfer Part (MTP) on the board to the user part on the host computer.

Messages will only be issued for user parts that were enabled during configuration.

5.4.5 MTP Pause Indication

Synopsis:

Primitive request issued by MTP3 to indicate the inability of providing the MTP service to the specified point code.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**MTP_MSG_PAUSE** (0x8403)
id		user_part_reference
src		**MTP_TASK_ID**
dst		User Part Module ID
rsp_req		0
hclass		0
status		0
err_info		0
len		4
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	4	dpc

Description:

This message is sent by the board to each user part that has been enabled in the event that there is no signalling route available to convey messages to the destination point code specified. On receipt of the message the user part should stop sending MSUs destined for the affected signalling point to the board until a subsequent MTP Resume Indication is received.

Parameter Description:

dpc - destination point code

The destination point code of the affected signalling point.

5.4.6 MTP Resume Indication

Synopsis:

Primitive request issued by MTP3 to indicate the ability of providing the MTP service to the specified point code.

Message Format:

MESSAGE HEADER		
FIELD NAME		**MEANING**
type		**MTP_MSG_RESUME** (0x8404)
id		user_part_reference
src		**MTP_TASK_ID**
dst		User Part Module ID
rsp_req		0
hclass		0
status		0
err_info		0
len		4
PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	4	dpc

Description:

This message is sent by the board to each user part that has been enabled in the event that a destination signalling point which was previously inaccessible becomes available. On receipt of the message the user part may resume sending MSUs destined for the specified signalling point to the board.

When the link set is activated the user should await reception of an MTP Resume Indication before starting traffic to each destination signalling point.

Parameter Description:

dpc - destination point code

The destination point code of the affected signalling point.

5.4.7 MTP Status Indication

Synopsis:

This primitive is used by level 3 to indicate to the user the partial inability of providing the MTP service to the specified destination.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MTP_MSG_STATUS** (0x8405)
id	user_part_reference
src	**MTP_TASK_ID**
dst	User Part Module ID
rsp_req	0
hclass	0
status	1 Remote User Unavailable 2 Signalling Network Congestion
err_info	0
len	6

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	4	Affected destination point code.
4	2	Congestion status (if status = 0x02).

Description:

This message is sent by MTP3 to each user part that has been enabled in the event that a destination signalling point becomes congested. It is also sent to the local user part in the event that the remote user part is unavailable.

5.5 Event Indication Messages

5.5.1 Board Status Indication

Synopsis:

Message sent to the application on completion of the reset and download sequence or on detection of a board failure.

NOTE: This message is not required when using the configuration utility s7_mgt.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**SSD_MSG_STATE_IND** (0x06a0)
id	board_id
src	**SSD_TASK_ID** (0x20)
dst	mgmt_id for SSD
rsp_req	0
hclass	0
status	0x60 - Reset successful 0x62 - Board failure
err_info	0
len	0

Description:

This message is used to convey the status of a board reset operation (whether success of failure) to the user.

5.5.2 LIU Status Indication

Synopsis:

Message issued by the board to notify of changes of LIU status.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MVD_MSG_LIU_STATUS** (0x0e01)
id	**liu_id**
src	**MVD_TASK_ID**
dst	**MGMT_TASK_ID**
rsp_req	0
hclass	0
status	**liu_status** (see below)
err_info	Reserved for future use
len	0

Description:

This message is issued by the board for every change of state on the trunk interface.

Parameter Description:

liu_id:
The identity of the Line Interface Unit to which the status indication applies. Set to 0 for line interface A or 1 for line interface B (if present).

liu_status The status field in the message header is coded as follows:

Value	Mnemonic	State
10	**LIUS_SYNC_LOSS**	Frame Sync Loss
11	**LIUS_IN_SYNC**	Frame Sync OK
12	**LIUS_AIS**	AIS Detected
13	**LIUS_AIS_CLRD**	AIS Cleared
14	**LIUS_REM_ALARM**	Remote Alarm
15	**LIUS_REM_ALM_CLRD**	Remote Alarm Cleared
20	**LIUS_PCM_LOSS**	PCM Loss
21	**LIUS_PCM_OK**	PCM Restored
22	**LIUS_FRAME_SLIP**	Frame Slip
25	**LIUS_BER5_OCRD**	BER > 1 in 100,000 (E1 only)
26	**LIUS_BER5_CLRD**	BER5 cleared
27	**LIUS_BER3_OCRD**	BER > 1 in 1,000 (E1 only)
28	**LIUS_BER3_CLRD**	BER3 cleared

5.5.3 Error Indication

Synopsis:

Message issued to management to advise of errors or unexpected events occurring within the protocol software.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MGT_MSG_EVENT_IND** (0x0008)
id	0 (unless shown below)
src	sending module id
dst	Management module id
rsp_req	0
hclass	0
status	ERROR CODE (see below)
err_info	Timestamp
len	0

The ERROR_CODE is coded as shown in the following table:

Value	Mnemonic	id	Description
0x31	S7E_RESET_ERR		MTP2 Failed to initialise.
0x33	S7E_POOL_EMPTY	l2_llid	No free buffers in MTP2 transmit pool.
0x34	S7E_TX_FAIL	l2_llid	Failed to send LSSU/FISU to driver.
0x35	S7E_HDR_ERR	l2_llid	No room to add level 2 header, SU not transmitted.
0x36	S7E_LEN_ERR	l2_llid	Length Error, SU not transmitted.
0x37	S7E_MSU_SEND	l2_llid	Failed to send SU to lower layer, protocol should handle retransmission.
0x39	S7E_BAD_PRIM	l2_llid	MTP2 unable to accept primitive.
0x3a	S7E_BAD_LLID	l2_llid	Invalid l2_llid in HDR structure.
0x3b	S7E_MEM_ERR	l2_llid	MTP2 memory allocation error.
0x3c	S7E_RTVL_ERR	l2_llid	MTP2 failure to perform retrieval.
0x51	MTP_BAD_PRIM	0	MTP3 unable to accept primitive.
0x52	MTP_POOL_EMPTY	0	No free frames in MTP3 transmit pool.
0x53	MTP_TX_FAIL	0	MTP3 failed to send MSU to lower layer.
0x54	MTP_LEN_ERR	0	MSU too long for buffer.
0x55	MTP_SLT_FAIL	link_id	Signalling link test failure.
0x57	MTP_TALLOC_ERR	0	MTP3 Failed to allocate T_FRAME.
0x58	MTP_BAD_ID	0	Invalid ID in message HDR.
0x59	MTP_MALLOC_ERR	0	MTP3 unable to allocate MSG.
0x5a	MTP_BSNT_FAIL	link_id	Failure to retrieve BSNT.
0x5b	MTP_RTV_FAIL	link_id	Retrieval failure.
0x5c	MTP_BAD_FSN	link_id	Erroneous FSN in COA.
0x5d	MTP_BAD_COO	link_id	COO received after changeover complete.
0x5e	MTP_SNMM_ERR	0	Internal software error.
0x5f	MTP_SLTM_ERR	0	Internal software error.
0x60	MTP_NO_COA	link_id	Failed to receive COA.
0x61	MTP_NO_CBA	link_id	Failed to receive CBA.
0x66	MTP_TIM_ERR	timer ref	MTP3 attempt to re-use active timer resource
0x67	MTP_RRT_OVRFLW		Messages discarded due to overflow of Re-Routing buffer.
0x68	MTP_FLUSH_FAIL	link_id	MTP3 failed to receive Flush Ack from level 2.
0x69	MTP_FLUSH_L2	link_id	MTP2 transmission buffers flushed (due to RPO).

5.5.4 MTP2 Level 2 State Indication

Synopsis:

Indication issued by the board every time the level 2 link state control state machine changes state.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MGT_MSG_SS7_STATE** (0x0201)
id	**llid** (Level 2 logical link id - 0, 1 or 2)
src	**SS7_TASK_ID**
dst	**MGMT_TASK_ID**
rsp_req	0
hclass	0
status	**LINK_STATE** (see below)
err_info	Reserved for future use
len	0

Description:

This message is issued by the SS7 Level 2 module on the board every time a change of state takes place at level 2. It is intended only for diagnostic use by system management. Normally the MTP Pause and MTP Resume Indications are used by the user parts to determine destination accessibility.

The level 2 link state control state machine is defined in Q.703.

Parameter Description:

LINK_STATE

The status field in the message header is used to indicate the state that has just been entered. It is coded as follows:

Value	Mnemonic	State
1	**S7S_IN_SERVICE**	In Service
2	**S7S_OUT_SERVICE**	Out of Service
3	**S7S_INIT_ALIGN**	Initial Alignment
4	**S7S_ALIGN_NOT_RDY**	Aligned, Not Ready
5	**S7S_ALIGN_READY**	Aligned, Ready
6	**S7S_PROC_OUTAGE**	Processor Outage

5.5.5 MTP2 Q.791 Event Indication

Synopsis:

Message issued by MTP2 to advise management of protocol events in accordance with Q.791.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MGT_MSG_SS7_EVENT** (0x0202)
id	l2_llid
src	MTP2 module id
dst	Management module id
rsp_req	0
hclass	0
status	EVENT CODE (see below)
err_info	Timestamp
next	0
len	0

Description:

This primitive is used by MTP2 to advise management of the occurance of protocol related events in accordance with Q.791. Currently these events either relate to the reason for a signalling link (that was in service) going out of service (events prefixed S7F_) or the occurrance of a congestion related event (prefixed S7G_).

The EVENT CODE is coded as shown in the following table:

Value	Mnemonic	Description
0	S7F_STOP	Stop request received
1	S7F_FIBR_BSNR	Abnormal FIBR/BSNR
2	S7F_EDA	Excessive delay of acknowledgement.
3	S7F_SUERM	Excessive error rate (SUERM).
4	S7F_ECONG	Excessive congestion.
5	S7F_SIO_RXD	Unexpected SIO received.
6	S7F_SIN_RXD	Unexpected SIN received.
7	S7F_SIE_RXD	Unexpected SIE received.
8	S7F_SIOS_RXD	SIOS received.
16	S7G_CONG	Onset of signalling link congestion.
17	S7G_CONG_CLR	Abatement of signalling link congestion.
18	S7G_CONG_DIS	Congestion event caused MSU discard.

5.5.6 MTP3 Q.791 Event Indication

Synopsis:

Message issued by MTP3 to notify management of various protocol events in accordance with CCITT Q.791.

Message Format:

MESSAGE HEADER	
FIELD NAME	**MEANING**
type	**MGT_MSG_MTP_EVENT** (0x0301)
id	0
src	MTP3 module id
dst	Management module id
rsp_req	0
hclass	0
status	EVENT CODE (see below)
err_info	Timestamp
len	Either 0, 1, 2 or 4

PARAMETER AREA		
OFFSET	**SIZE**	**NAME**
0	len	Event specific parameters

The EVENT_CODE coding and the meaning of the event specific parameters are given in the following table:

link is indicated as (linkset_id * 256) + link_ref, (size = 2).
linkset is indicated as linkset_id, (size = 1).
point code is a 4 byte value, (size = 4).

Value	Mnemonic	Parameter	Description
1	MTPEV_CO	link	Changeover
2	MTPEV_CB	link	Changeback
3	MTPEV_REST	link	Restoration commenced
4	MTPEV_RPO	link	Remote processor outage
5	MTPEV_RPO_CLR	link	Remote processor outage cleared
6	MTPEV_CONG	link	Signalling link congestion
7	MTPEV_CONG_CLR	link	Congestion cleared
8	MTPEV_CONG_DIS	link	MSU discarded due to congestion
9	MTPEV_LS_LOST	linkset	Link set failure
10	MTPEV_LS_OK	linkset	Link set recovered
13	MTPEV_DEST_LOST	point code	Destination unavailable
14	MTPEV_DEST_OK	point code	Destination available
15	MTPEV_AJSP_LOST	linkset	Adjacent SP inaccessible
16	MTPEV_AJSP_OK	linkset	Adjacent SP accessible.

APPENDIX A: PROTOCOL CONFIGURATION COMMANDS

This appendix describes the commands and parameters used in the protocol configuration file config.txt. These are used by the s7_mgt utility to perform single-shot configuration of the protocol stack at startup.

A1 Physical Interface Parameters

A1.1 PCCS3 Board Configuration

Syntax:PCCS3_BOARD <pcm_id> <board_id> <num_pcm> <flags> <code_file>

Example: PCCS3_BOARD 0 0 1 0x0003 mtp73.dc1

Configuration command to install a PCCS3 board in the system.

<pcm_id>

The logical id of the first on-board pcm port within the system as a whole.

<board_id>

The logical id of the board within the system in the range 0 .. 15.

<num_pcm>

The number of PCM ports on the card. If this parameter is set to zero, then the default for the card will be assumed, i.e. 1 for the PCCS3.

<flags>

A 16 bit value that provides additional level 1 configuration for the board. The meaning of each bit may vary with different board types. For the PCCS3 the bits in the flags field are used as follows:

Bit 0 is set to 1 to select an external clock reference or zero to use the internal clock source. When using external clock reference the clock is recovered from the E1/T1 interface.

Bit 1 is set to 1 to cause a signalling link to be taken out of service if no signalling units are received for an excessive period. If this bit is set to zero then the link will remain in service if continuous flags are received (i.e. no FISUs). Usually this bit should be set to 1.

Bit 11 is used to select one of two possible 56kbit/s operating modes. It is only effective when the per-link 56kbit/s option is selected and the signalling link is not running on a serial port. When set to 1, 56kbit/s operation does not use bit 8 in the timeslot and when set to zero 56kbit/s operation does not use bit 7 in the timeslot. (The bits of the timeslot are labelled bit 1 .. bit 8 in accordance with ITU-T terminology). Usually this bit should be set to 1.

All other bits are reserved and should be set to zero.

<code file>

The name of the code file which gets downloaded to the board when it is reset. Code files for the PCCS3 are all supplied with the suffix .dc1. The main part of the filename depends on which protocol modules are included. Examples are: mtp73.dc1, isup73.dc1, tup73.dc1.

A1.2 Septel ISA (PCCS6) Board Configuration

Syntax: PCCS6_BOARD <pcm_id> <board_id> <num_pcm> <flags> <code_file>

Example: PCCS6_BOARD 0 0 2 0x00c5 mtp76.dc2

Configuration command to install a PCCS6 board in the system.

<pcm_id>

The logical id of the first on-board pcm port within the system as a whole.

<board_id>

The logical id of the board within the system in the range 0 .. 15.

<num_pcm>

The number of PCM ports on the card. If this parameter is set to zero, then the default for the card (ie 2) is assumed.

<flags>

A 16 bit value that provides additional level 1 configuration for the board. The meaning of each bit may vary with different board types. For the Septel ISA (PCCS6) the bits in the flag are used as follows:

Bit 0 is set to 1 to select an external clock reference (as recovered from a PCM interface) or zero to use the internal clock source. When set to 1 the setting of bit 2 determines which LIU is used to recover clocks.

Bit 1 is set to 1 to cause a signalling link to be taken out of service if no signalling units are received for an excessive period. If this bit is set to zero then the link will remain in service if continuous flags are received (i.e. no FISUs). Usually this bit should be set to 1.

Bit 2 is only significant when bit 0 is set to 1. It should be set to zero to recover clock from the first PCM port, or 1 to recover clocks from the second PCM port.

Bit 4 is set to 1 for 2MHz SCbus operation, or zero to select 4MHz operation. The use of a 4MHz clock is recommended as it offers twice as many channels on the SCbus.

Bit 6 and **7** together select the SCbus mode according to the following table:

Bit 7	Bit 6	SCbus mode
0	0	The SCbus interface is disabled. This mode should be selected whenever the SCbus module is not fitted or when the user does not wish to use the SCbus.
0	1	**Master** - The board will drive the SCbus clocks.
1	0	**Standby** - The board is configured to take over as SCbus clock master in the event that the SCbus clocks fail.
1	1	**Slave** - The board uses the SCbus clocks (which must be generated by another board on the SCbus).

Bit 11 is used to select one of two possible 56kbit/s operating modes. It is only effective when the per-link 56kbit/s option is selected and the signalling link is not running on a serial port. When set to 1, 56kbit/s operation does not use bit 8 in the timeslot and when set to zero 56kbit/s operation does not use bit 7 in the timeslot. (The bits of the timeslot are labelled bit 1 .. bit 8 in accordance with ITU-T terminology). Usually this bit should be set to 1.

All other bits are reserved and should be set to zero.

<code file>

The name of the code file which gets downloaded to the board when it is reset. Code files for Septel ISA are all supplied with the suffix .dc2. The main part of the filename depends on which protocol modules are included. Examples are: mtp76.dc2, isup76.dc2, tup76.dc2.

A1.4 LIU_CONFIG Command

Syntax:LIU_CONFIG <board_id> <liu_id> <liu_type> <line_code> <frame_format> <crc_mode>

Example:

This command is used during initialisation to configure the operating parameters for an E1 / T1 line interface unit.

<board_id>

The logical identity of the board in the range from 0 to one less than the number of boards supported.

<liu_id>

The identifier of the E1/T1 Line Interface Unit. This should be set to 0 for the first LIU and 1 for the second LIU.

<liu_type>

The physical type of interface according to the following table: (note that this must be selected by the user to be appropriate for the actual hardware fitted otherwise an error status will be returned).

liu_type	Description
1	Disabled (used to deactivate a LIU). In this mode the LIU will not produce an output signal.
2	E1 75ohm unbalanced interface
3	E1 120ohm balanced interface
4	T1
5	E1 75ohm or 120ohm setting based on fitted hardware. (This is the preferred method of selecting an E1 interface)

<line_code>

The line coding technique taken from the following table:

line_code	Description
1	HDB3 (E1 only) [E1 default]
2	AMI with no Zero Code Suppression
3	AMI with Zero Code Suppression (The appropriate bit in the clear_mask parameter may be set to disable Zero Code Suppression for individual timeslots if required) (T1 only)
4	B8ZS (T1 only) [T1 default]

<frame_format>

The frame format taken from the following table:

frame_format	Description
1	E1 double frame (E1 only) [E1 default]
2	E1 CRC4 multiframe (E1 only)
3	F4 (T1 only)
4	D3/D4 (Yellow alarm = bit 2 in each channel) (T1 only) [T1 default]
5	D3/D4 (Yellow alarm = last bit of multiframe) (T1 only)
6	ESF (Yellow alarm = bit 2 in each channel) (T1 only)
7	ESF (Yellow alarm in data link channel) (T1 only)
8	SLC 96 (T1 only)

<crc_mode>

The CRC mode taken from the following table:

crc_mode	Description
1	CRC generation disabled [E1 default][T1 default]
2	CRC4 enabled (E1 only)
3	CRC4 compatibility mode (E1 only)
4	CRC6 enabled (T1 only)

A1.5 LIU_SC_DRIVE Command

Syntax: LIU_SC_DRIVE <board_id> <liu_id> <sc_channel> <ts_mask> {<mode>}

Example: LIU_SC_DRIVE 0 0 512 0xfffefffe

This command is used during initialisation to set up a static switch path through the card between the Line Interface Unit (LIU) and the SCbus. It connects selected incoming voice timeslots from an E1/T1 LIU to a sequential block of channels on the SCbus and prepares the outgoing timeslots for subsequent use by the MVD_MSG_SC_LISTEN message.

<board_id>

The logical identity of the board in the range from 0 to one less than the number of boards supported.

<liu_id>

The identifier of the E1/T1 Line Interface Unit. This should be set to 0 for the first LIU and 1 for the second LIU.

It is also possible to use this command to connect the signalling processor timeslots to the SCbus. Contact DataKinetics for more information about this mode of operation.

<sc_channel>

The channel number of the first channel to be used on the SCbus. This should be in the range from 0 up to one less than the total number of channels on the SCbus. In the case of a 4MHz SCbus the valid range for sc_channel is 0 .. 1023.

When deciding what value to use for sc_channel the user should note that Dialogic cards usually allocate channels on the SCbus commencing at the lowest numbered channels and working up. It is therefore recommended that the user should determine the maximum number of sc_channels that will be used by any Dialogic boards in the system (allowing for future expansion) and set the sc_channel to the next value.

<ts_mask>

A 32 bit timeslot mask where each bit position is set to 1 if the corresponding timeslot on the E1/T1 interface is required to be connected to the SCbus. The least significant bit (bit 0) represents timeslot 0. Each timeslot for which the corresponding bit is set in ts_mask will be connected up to the SCbus; other timeslots will not be affected in any way.

Timeslots containing SS7 signalling that will be processed by the signalling processor on the card should not be included in the timeslot mask. Usually the mask should be set to include all bearer (voice) timeslots but no signalling timeslots. Bit 0 (corresponding to timeslot 0 on the LIU) will not usually be set as timeslot 0 for an E1 interface contains synchronisation information whilst timeslot 0 for a T1 interface does not even exist.

As an example, for an E1 interface with SS7 signalling on timeslot 16, and the remaining 30 timeslots used for voice circuits, ts_mask should be set to the value 0xfffefffe. For a T1 interface with signalling on timeslot 24, ts_mask should be set to the value 0x00fffffe

<mode>

This parameter allows the user to select how the SCbus channels are allocated. Usually (mode=1) the first timeslot connected to the SCbus is connected to sc_channel and each subsequent timeslot that is connected will be connected to the next SCbus channel. This allows maximum utilisation of channels on the SCbus and is compatible with channel allocation by Dialogic cards.

An alternative mode (mode=2) (that should only be used if there is a specific requirement for it) associates (but does not necessarily connect) timeslot 0 on the LIU with sc_channel and subsequent timeslots on the LIU with subsequent SCbus channels. Connections are only made when the corresponding bit in the timeslot mask is set to 1. This mode of operation preserves the spacing between timeslots that was originally found on the E1/T1 interface but does result in a number of SCbus channels being not used.

The mode parameter is optional and may be omitted altogether. This will have the same effect as setting it to 1.

A1.6 SCBUS_LISTEN Command

Syntax: SCBUS_LISTEN <board_id> <liu_id> <timeslot> <sc_channel>

Example: SCBUS_LISTEN 0 0 31 23

This command establishes a connection from the SCbus to an outgoing timeslot on the Septel ISA card Line Interface Unit (LIU).

Typically the SCBUS_LISTEN command is used for two purposes; to ensure that all transmit timeslots are set to generate the appropriate voice idle pattern and to set up any semi-permanent connections between the SCbus and the LIU.

Correct operation of the command is dependent upon the prior use of the LIU_SC_DRIVE command.

Dynamic modification of voice paths can only be performed by issuing messages directly to the board. The MVD_MSG_SC_LISTEN message is recommended for this purpose.

<board_id>

The logical identity of the board in the range from 0 to one less than the number of boards supported.

<liu_id>

The identifier of the E1/T1 Line Interface Unit. This should be set to 0 for the first LIU and 1 for the second LIU.

<timeslot>

The timeslot number on the E1/T1 line interface unit on which the data from the SCbus will be transmitted. The valid range for **timeslot** is 1 to 31 for an E1 interface and 1 to 24 for a T1 interface. The timeslot must have been enabled by setting the appropriate bit in the **ts_mask** parameter of the LIU_SC_DRIVE command

<sc_channel>

The channel number on the SCbus to which the LIU will listen. This should be in the range from 0 up to one less than the total number of channels on the SCbus. In the case of a 4MHz SCbus the valid range for sc_channel is 0 .. 1023.

A1.7 SCBUS_PATTERN Command

Syntax: SCBUS_PATTERN <board_id> <pattern_id> <sc_channel> <pattern>

Example: SCBUS_PATTERN 0 0 512 0x2a

This command uses one of the pattern generators on the Septel ISA board to generate a user-defined pattern on a specified SCbus channel. This pattern can then be connected to any outgoing LIU timeslots using either the SCBUC_LISTEN command or the MVD_MSG_SC_LISTEN message.

Typically the SCBUS _PATTERN is used to generate a voice idle pattern that can then be used as the source information for transmission on any LIU timeslots that do not contain an active call.

<board_id>

The logical identity of the board in the range from 0 to one less than the number of boards supported.

<pattern_id>

The logical identifier for the pattern generator used to generate the pattern. This must be in the range 0 to 31 and would usually be set to 0 for the first pattern, 1 for the next and so on.

<sc_channel>

The channel number on the SCbus where the pattern will be generated. This should be in the range from 0 up to one less than the total number of channels on the SCbus. In the case of a 4MHz SCbus the valid range for sc_channel is 0 .. 1023.

<pattern>

The value of the pattern data to be generated on the SCbus. The value should be in the range 0 to 255. Typically the value might be 0xff for an all ones idle pattern or 0x2a for an ITU-T E1 idle pattern.

A2 MTP Parameters

A2.1 Global MTP configuration

Syntax: MTP_CONFIG <reserved1> <reserved2> <options>

Example: MTP_CONFIG 123 8 0

The global configuration parameters for the Message Transfer Part.

<reserved1>, <reserved2>

These parameters are reserved for backwards compatibility. For applications conforming to this release of the documentation they should always be set to zero.

<options>

A 32 bit value containing run-time options for the operation of MTP level 3. Bits 0, 1, 8, 9, 10 and 11 have the same meaning as the corresponding bits in the l3_flags field of the Board Configuration Request message described in an earlier section.

A2.2 MTP Link Set

Syntax: MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags> <local_spc> <ssf>

Example: MTP_LINKSET 0 321 2 0x0000 456 0x08

Configuration of a link set to an adjacent signalling point.

<linkset_id>

The logical identity of the link set, in the range 0 to one less than the number of link sets supported, The linkset_id is used in other commands for reference.

<adjacent_spc>

The signalling point code of the adjacent signalling point.

<num_links>

The (maximum) number of links that will be allocated to the link set.

<flags>

A 16 bit value reserved for future use to specify run time options. Currently no options are defined so the parameter must be set to zero.

<local_spc>

The signalling point code of the signalling point itself.

<ssf>

The value to be used in the sub-service field of all level 3 messages and checked for by the discrimination function in all received messages. This is a 4 bit value. Note that for ANSI operation the two least significant bits should be set to 1.

NOTE: For correct operation the adjacent point code must also appear in a MTP_ROUTE declaration.

A2.3 MTP Signalling Link

Syntax: MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink> <stream> <timeslot> <flags>

Example: MTP_LINK 0 0 2 2 0 1 16 16 0x0006

Configuration of a signalling link.

<link_id>

The link's unique logical link identity. It must be in the range 0 to one less than the total number of signalling links supported.

<linkset_id>

The logical identity of the link set to which the link belongs. The linkset must already have been configured using the MTP_LINKSET command.

<link_ref>

The logical identity within the link set of the signalling link. It should be in the range 0 to one less than the number of links in the link set.

<slc>

The signalling link code for the signalling link. This must be unique within the link set and will usually be the same as the <link_ref>. The valid range is 0…15.

<board_id>

The board id of the signalling processor allocated for this signalling link.

<blink>

The index of the signalling processor (within the board) allocated for this signalling link. It should be in the range 0 to one less than the number of signalling processors on the board.

<stream>

A reference to the logical 2Mb/s PCM highway from which the signalling processor is to insert the signalling. Typically this is set to 16 to recover signalling from the first PCM port on the board. For full information on the board data paths, refer to the board User Manual.

<timeslot>

The timeslot on the <stream> which should be used for signalling in the range is 0 .. 31. For an E1 interface the valid range is 1 .. 31. For a T1 interface the valid range is 1 .. 24.

<flags>

A 16 bit value containing additional run-time options.

Bit 0 is set to 1 to force the use of the emergency proving period during link alignment or 0 to use the appropriate proving period according to Q.703.

Bit 1 is set to 1 to cause a signalling link test (in accordance with ITU-T Q.707) to be carried out before a link is put into service, or 0 if a test is not required.

Bit 2 is set to 1 to cause a signalling link test (in accordance with ITU-T Q.707) to be carried out every 30 seconds. Note that this bit is ignored unless bit 1 is also set to 1.

Bit 8 is used to select the MTP2 error correction mode. It is set to 1 to select PCR (Preventive Cyclic Retransmission) operation or zero for the Basic Method of Error Correction.

Bit 11 is set to 1 to select 56kbit/s operation for the link or 0 for 64kbit/s operation.

NOTE: When using a serial port, 56kbit/s operation is only supported when the clock is applied externally.

Bit 13 is only used when the link has been configured to run over a serial port. If set to 1 an external clock will be used (Receive clock). If set to zero an internal clock (Transmit clock) will be used. If the link has not been configured to run over a serial port, this bit must be set to 0.

Bit 14 is set to 1 to use a serial port rather than a PCM timeslot for this link. In this mode the stream and timeslot parameters for this link will be ignored (and should be set to zero). If this bit is set to zero, the link will use the specified stream and timeslot. The serial port used by the signalling processors for each link is fixed, according to the following table:

blink	Serial Port
0	Cannot be used for a serial port
1	A
2	B

Bit 15 is set to 1 to disable the link or 0 to enable the link.

All other bits are reserved for future use and should be set to zero.

A2.4 MTP Route

Syntax: MTP_ROUTE <dpc> <norm_ls> <user_part_mask> <flags> <second_ls>

Example: MTP_ROUTE 567 0 0x0020 0x0000 0

Configuration of a route for use with one or more user parts.

<dpc>

The point code of the remote signalling point for which this command is configuring routing data. It may be either an adjacent point code or a point code accessible via an adjacent Signalling Transfer Point..

<norm_ls>

The linkset_id of the normal link set used to reach the specified destination. The norm_ls must be a linkset_id that has already been configured using the MTP_LINKSET command. The normal link set may be any of the following:

a) The only link set used to reach the destination.
b) The preferred link set used to reach the destination.
c) One of a pair of links sets forming a combined link set.

In the latter two cases a second link set (second_ls) must also be specified.

Within a link set messages will automatically be load shared across links using the Signalling Link Selection (SLS) field in the message.

<second_ls>

The linkset_id of an optional second link set used to reach the specified destination. This may be either of the following options:

a) The secondary link set used to reach the destination only on failure of the preferred link set.
b) One of a pair of links sets forming a combined link set over which load sharing will take place. (in this case bit 1 must also be set in the <flags> parameter of the command).

When a second link set is specified the user must also set **bit 0** in the <flags> field of this command.

<user_part_mask>

This is a 16 bit field used identify the user parts that are supported over this route. The bits are labelled 0 to 15 and for each user part supported the bit corresponding to the Service Indicator for that user part should be set. (eg. To support just ISUP messages, the ISUP Service Indicator is 5 so bit 5 should be set. Therefore a value of 0x0020 would be appropriate).

<flags>

A 16 bit field containing run-time configuration options for the route as follows:

Bit 0 is set to 1 to indicate that a second link set is specified within the command. If zero the second_ls parameter is ignored.

Bit 1 is used to determine whether or not to load share messages across the two link sets. It is only used when two link sets are specified for the route. When set the MTP3 module will load share messages for the destination equally across each of the two specified link sets. Otherwise the MTP3 module will consider the normal link set to be the preferred link set and will only use the second link set in the event of failure of the normal link set. The bit should be set to 1 to enable load sharing across the two link sets or zero to disable load sharing and use preferred and secondary link sets.

All other bits are reserved for future use and must be set to zero.

A2.5 MTP User Part

Syntax: MTP_USER_PART <si> <module_id>

Example: MTP_USER_PART 0x0a 0x2d

Configuration of a local user part module (other than a user part which has its own config command in config.txt)

<si>

The service indicator.

<module_id>

The module id of the user process.

NOTE: This command must not be used when the corresponding configuration commands are used (ISUP_CONFIG, TUP_CONFIG, NUP_CONFIG etc) as these commands automatically perform the function of the MTP_USER_PART command.

A3 ISUP Parameters

A3.1 Global ISUP Configuration

Syntax: ISUP_CONFIG <res1> <res2> <res3> <options> <num_grps> <num_ccts>

Example: ISUP_CONFIG 0 0 0 0x0001 4 128

The global configuration parameters for the ISUP User Part module.

<res1>, <res2>, <res3>

These parameters are reserved for backwards compatibility. For applications conforming to this release of the documentation they should always be set to zero.

<options>

A 16 bit value containing global run-time options for the operation of the ISUP module. The meaning of each bit is as defined for the 'options' parameter in the ISUP Configure Request message as detailed in the *System7 ISUP Programmer's Manual*

<num_grps>

The maximum number of ISUP circuit groups that the user intends to use. This must not exceed the maximum number of circuit groups supported otherwise module configuration will fail. Typically <num_grps> should be set to the maximum number of circuit groups supported.

<num_ccts>

The maximum number of ISUP circuits that the user intends to use. This must not exceed the maximum number of circuits supported otherwise module configuration will fail. Typically <num_ccts> is set to 32 times the number of groups for E1 operation and 24 times the number of circuit groups for T1 operation.

NOTE: The valid range for the circuit identifier (cid) is from zero up to one less than the maximum number of circuits (num_ccts).

A3.2 ISUP Circuit Group Configuration

Syntax: ISUP_CFG_CCTGRP <gid> <dpc> <cic> <cid> <cic_mask> <options>
<user_inst> <user_id> <opc> <ssf> <variant> <options2>

Example: ISUP_CFG_CCTGRP 0 3 1 1 0x7fff7fff 0x00000003

The configuration parameters for a group of ISUP circuits.

<gid >

The group id of the circuit group in the range 0 to one less than the number of groups supported.

<dpc>

The destination point code for all circuits in the circuit group.

<base_cic>

The Circuit Identification Code (CIC) that will be allocated to the first circuit in the circuit group.

<base_cid>

The logical id for the first circuit in the circuit group. It must lie in the range 0 to one less than the number of circuits supported.

<cic_mask>

A 32 bit mask with bits set to indicate which circuits are to be allocated to the circuit group. Bit zero must always be set as it represents the base_cic / base_cid. Subsequent bits represent subsequent circuits.

<options1>

A 32 bit value containing run-time options for the ISUP circuit group. This 32 bit field is made up of two 16 bit fields as described in the 'Configure Circuit Group Request' section of the ISUP Programmers Manual. The least significant 16 bits are the **options** and the most significant 16 bits the **ext_options**. Note also the <options2> parameter for this command.

<user_instance>

The instance number of the application using this circuit group. Usually only a single instance is present so this should be set to zero.

<user_id>

The module_id of the application using this circuit group.

<opc>

Originating Point Code. The local point code for this circuit group.

<ssf>

The sub-service field value for all messages relating to this circuit group.

<variant>

The variant of the ISUP protocol for this circuit group, as detailed in the *ISUP Programmer's Manual.*

<options2>

A 32 bit value containing additional run-time options for the ISUP circuit group as described for the **ext_1_options** field in the 'Configure Circuit Group Request' section of the ISUP Programmers Manual.

A4 TUP Parameters

A4.1 Global TUP Configuration

Syntax: TUP_CONFIG <res1> <res2> <res3> <options> <num_grps> <num_ccts>

Example: TUP_CONFIG 0 0 0 0x0000 4 128

The global configuration parameters for the TUP User Part module. This command activates the TUP User Part to run.

<res1>, <res2>, <res3>

These parameters are reserved for backwards compatibility. For applications conforming to this release of the documentation they should always be set to zero.

<options>

A 16 bit value containing global run-time options for the operation of the TUP module. The meaning of each bit is as defined for the 'options' parameter in the TUP Configure Request message as detailed in the *System7 TUP Programmer's Manual*

<num_grps>

The maximum number of TUP circuit groups that the user intends to use. This must not exceed the maximum number of circuit groups supported otherwise module configuration will fail. Typically <num_grps> should be set to the maximum number of circuit groups supported.

<num_ccts>

The maximum number of TUP circuits that the user intends to use. This must not exceed the maximum number of circuits supported otherwise module configuration will fail. Typically <num_ccts> is set to 32 times the number of groups for E1 operation and 24 times the number of circuit groups for T1 operation.

NOTE: The valid range for the circuit identifier (cid) is from zero up to one less than the maximum number of circuits (num_ccts).

A4.2 TUP Circuit Group Configuration

Syntax: TUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options> <user_inst> <user_id> <opc> <ssf>

Example: TUP_CFG_CCTGRP 0 3 1 1 0x7fff7fff 0x0003

The configuration parameters for a group of TUP circuits.

<gid >

The group id of the circuit group in the range 0 to one less than the number of groups supported.

<dpc>

The destination point code for the circuits in the circuit group.

<base_cic>

The Circuit Identification Code (CIC) that will be allocated to the first circuit in the circuit group.

<base_cid>

The logical id for the first circuit in the circuit group. It must lie in the range 0 to one less than the number of circuits supported.

<cic_mask>

A 32 bit mask with bits set to indicate which circuits are to be allocated to the circuit group. Bit zero must always be set as it represents the base_cic/base_cid. Subsequent bits represent the subsequent circuits.

<options>

A 32 bit value containing run-time options for the TUP circuit group (see 'Configure Circuit Group Request' section of the *System7 TUP Programmers Manual*).

<user_instance>

The instance number of the application using this circuit group. Usually only a single instance is present so this should be set to zero.

<user_id>

The module_id of the application using this circuit group.

<opc>

Originating Point Code. The local point code for this circuit group.

<ssf>

The sub-service field value for all messages relating to this circuit group.

A5 NUP Parameters

A5.1 Global NUP Configuration

Syntax: NUP_CONFIG <res1> <res2> <res3> <options> <num_grps> <num_ccts>

Example: NUP_CONFIG 0 0 0 0xdde3 4 128

The global configuration parameters for the NUP User Part module. This command activates the NUP User Part to run.

<res1>, <res2>, <res3>

These parameters are reserved for backwards compatibility. For applications conforming to this release of the documentation they should always be set to zero.

<options>

A 16 bit value containing global run-time options for the operation of the NUP module. The meaning of each bit is as defined for the 'options' parameter in the NUP Module Configuration Request message as detailed in the *System7 NUP Programmer's Manual*

<num_grps>

The maximum number of NUP circuit groups that the user intends to use. This must not exceed the maximum number of circuit groups supported otherwise module configuration will fail. Typically <num_grps> should be set to the maximum number of circuit groups supported.

<num_ccts>

The maximum number of NUP circuits that the user intends to use. This must not exceed the maximum number of circuits supported otherwise module configuration will fail. Typically <num_ccts> is set to 32 times the number of groups.

NOTE: The valid range for the circuit identifier (cid) is from zero up to one less than the maximum number of circuits (num_ccts).

A5.2 NUP Circuit Group Configuration

Syntax: NUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options> <user_inst> <user_id> <opc> <ssf>

Example: NUP_CFG_CCTGRP 0 3 1 1 0x7fff7fff 0x2503

The configuration parameters for a group of NUP circuits.

<gid >

The group id of the circuit group in the range 0 to one less than the number of groups supported.

<dpc>

The destination point code for the circuits in the circuit group.

<base_cic>

The Circuit Identification Code (CIC) that will be allocated to the first circuit in the circuit group.

<base_cid>

The logical id for the first circuit in the circuit group. It must lie in the range 0 to one less than the number of circuits supported.

<cic_mask>

A 32 bit mask with bits set to indicate which circuits are to be allocated to the circuit group. Bit zero must always be set as it represents the base_cic/base_cid. Subsequent bits represent the subsequent circuits.

<options>

A 16 bit value containing run-time options for the NUP circuit group (see 'Configure Circuit Group Request' section of the *System7 NUP Programmer's Manual*).

<user_instance>

The instance number of the application using this circuit group. Usually only a single instance is present so this should be set to zero.

<user_id>

The module_id of the application using this circuit group.

<opc>

Originating Point Code. The local point code for this circuit group.

<ssf>

The sub-service field value for all messages relating to this circuit group. For correct NUP operation this should usually be set to 8.

APPENDIX B: LIBRARY FUNCTION REFERENCE

Additional library functions are listed in the System7 Software Environment Programmer's Manual.

B1.1 rpackbytes

Synopsis:

Function to pack bytes into machine independent format.

Prototype:

```
void rpackbytes(unsigned char *dest, int dest_byte_offset,
                unsigned long value, int bytecount);
```

Return Value:

None.

Parameters:

dest - pointer to the destination buffer

dest_byte_offset - offset from the start of the destination buffer to store data

value - the value to be put into the buffer

bytecount - the number of significant bytes to take from value.

Description:

Packs the requested number of bytes into a buffer in a machine independent manner for sending to another module, regardless of byte ordering on either processor.

Example:

rpackbytes(dest, 10, value, 2);

This call will use the least significant 2 bytes of the value and store the resulting data starting at location dest + 10. The least significant byte of value will be written to dest + 11 and the next most significant byte to dest + 10.

B1.2 runpackbytes

Synopsis:

Function to extract bytes from machine independent format.

Prototype:

```
unsigned long runpackbytes(unsigned char *src, int src_byte_offset,
                           int bytecount);
```

Return Value:

The numeric value unpacked from the buffer.

Parameters:

src - pointer to the source buffer

src_byte_offset - offset from the start of the message buffer to retrieve data

bytecount - the number of bytes to take from the message.

Description:

Unpacks the requested number of bytes from a buffer, regardless of byte order on the processor.

Example:

result = (u16)runpackbytes(src, 10, 2);

This call will retrieve the least two significant bytes from the buffer src and return the value as a u16. The u16 will be formed by src + 11 as the least significant byte and src + 10 as the most significant byte.

Note : The user should cast the return value to the required type.